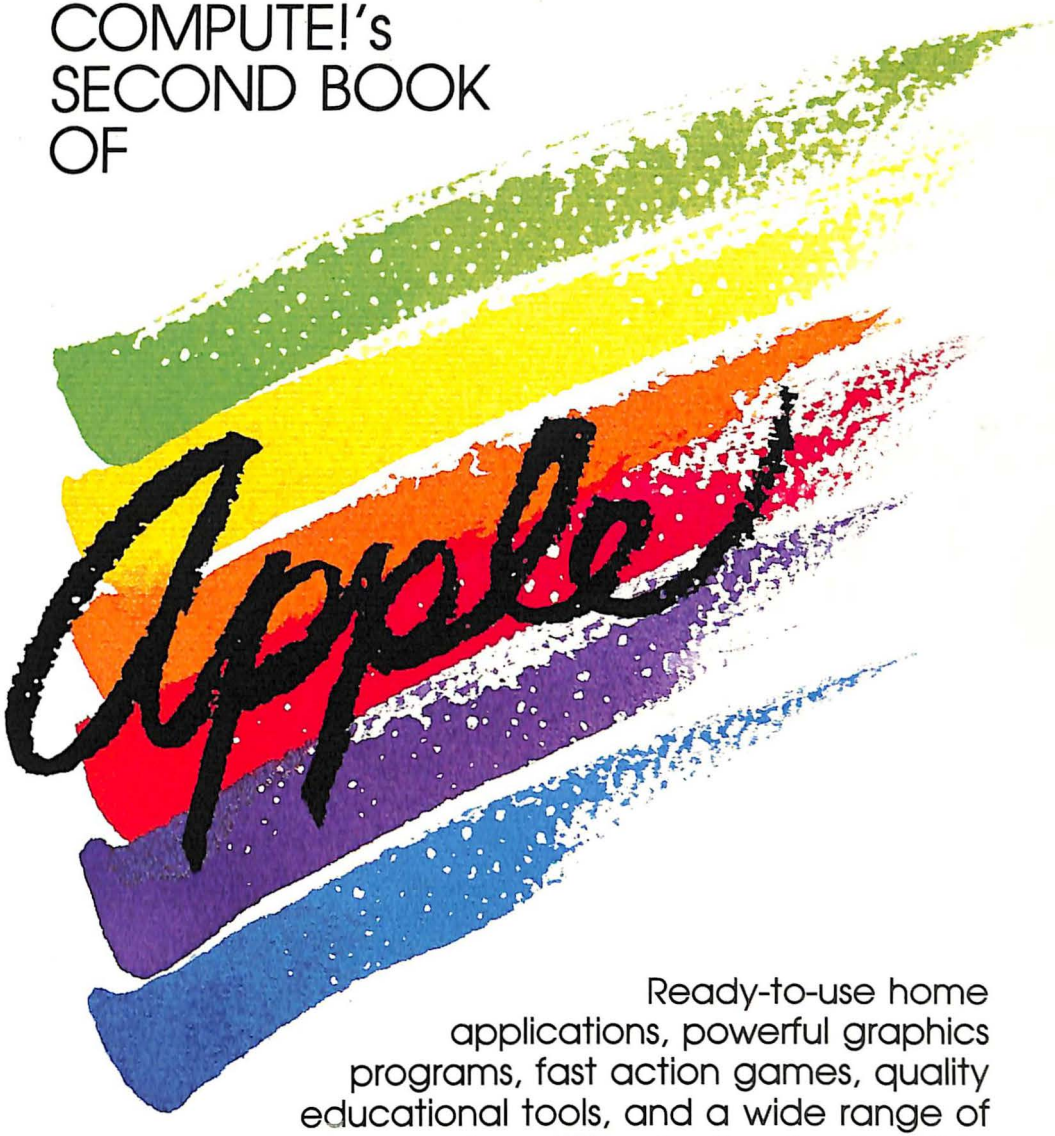
COMPUTE!'s SECOND BOOK OF



Ready-to-use home applications, powerful graphics programs, fast action games, quality educational tools, and a wide range of programming utilities for the Apple II, II+, IIe, and IIc computers.

A **COMPUTE! Books** Publication $12.95

COMPUTE!'s Second Book of APPLE

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies
Greensboro, North Carolina

The following articles were originally published in *COMPUTE!* magazine, copyright 1984, COMPUTE! Publications, Inc.: "3-D Drawing Master (February—originally titled "Commodore 3-D Drawing Master"); Function Keys for the Apple (April); "Applesoft Lister" (July); "Lightning Sort" (September); "Missile Math" (September); "Apple Screen Dump" (October); "Reflection" (November); "Softsearcher" (December—originally titled "Applesoft Searcher").

The following articles were originally published in *COMPUTE!* magazine, copyright 1985, COMPUTE! Publications, Inc.: "Paratrooper" (January); "Random Access DATA Statements for the Apple" (January); "Apple Bowling Champ" (February); "Apple SuperFont" (April); "Mindbusters" (April); "Apple IIc RAM Disk Mover" (May/June); "Home Financial Calculator" (May); "Space Dodger" (May); Apple Universal Input" (June); "Apple Variable Lister" (June—originally titled Apple ProDOS Variable Lister"); "Webster Dines Out" (June); "Apple Automatic Proofreader" (July); "Fast Filer" (July); "Softball Statistics" (July); "Apple Fractals" (September).

The following articles were originally published in *COMPUTE!'s Apple Special: Applications Issue*, copyright 1985, COMPUTE! Publications, Inc.: "AppleMLX" (Spring/Summer); "Chess" (Spring/Summer); "Heat Seeker" (Spring/Summer).

The following articles were orginally published in *COMPUTE!'s Apple Games for Kids*, copyright COMPUTE! Publications, Inc.: "Build a Quiz"; "Starving Artist."

The following article was orginally published in *COMPUTE!'s Guide to Telecomputing on the Apple*: "QUICK.BUT.DUMB."

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-87455-008-4

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is one of the ABC Publishing Companies and is not associated with any manufacturer of personal computers. Apple, Apple II, Apple II+, Apple IIe, Apple IIc, DOS 3.3, and ProDOS are trademarks of Apple Computer, Inc.

Contents

Foreword

Hundreds of thousands of people enjoy their Apple home computers. Over 400,000 Apple IIc computers became a part of American households in the last year alone. With so many Apple II, II+, IIe, and IIc computers out there, it's no wonder the games, educational programs, and financial, graphics, and programming applications written for the Apple are so popular.

COMPUTE!'s Second Book of Apple continues in the tradition of *COMPUTE!'s First Book of Apple* by bringing more high-quality, low-cost software to Apple computer owners. More than two dozen of the best Apple programs ever published in *COMPUTE!* magazine, as well as programs never before published, are included in this one package.

Machine language arcade games like "Heat Seeker" and "Webster Dines Out" rival commercial software in speed and graphics. "Apple Bowling Champ" puts you on the alley, going for that 300 game. And "Fill-In-the-Blank" gives you the framework to write your own adventure games.

Do you want your children to learn while they play? "Missile Math" teaches elementary mathematics in the midst of an exciting shoot-em-up game. "Starving Artist" lets children draw on the Apple screen. "Build a Quiz" lets them create their own quizzes on any subject.

Adults aren't neglected, either, for there are games of logic and thought that entertain for hours. "Chess," an excellent five-level computer chess game, pits you against a master opponent. "Mindbusters" challenges your perception and your patience. "Reflection" gives you an electronic reversi game, where the final outcome is often in doubt until the last play.

What about using your Apple for household duties? "Home Financial Calculator" lets you analyze almost every loan and investment situation. "Fast Filer" offers a simple-to-use database which you can use for filing anything from magazine articles to household possessions. And "QUICK.BUT.DUMB," a short terminal program, lets you access commercial information services and electronic bulletin boards with your modem.

Graphics generators are also a significant part of *COMPUTE!'s Second Book of Apple*. "Apple SuperFont," one of the most sophisticated character creators for the Apple ever published, lets you design your own character sets, even game characters. "3-D Drawing Master" and "Automatic Scaling Plotter" assist while you draw three-dimensional objects on the screen and produce scatter plots with your own data.

Programming utilities like "Applesoft Lister" and "Function Keys for the Apple" allow you to customize your program listings or put powerful commands at your fingertips. "Apple IIc RAM Disk Mover" and "Lightning Sort" let you use the IIc's memory to speed up programs or provide a way to sort a thousand items in less than three seconds.

COMPUTE!'s Second Book of Apple contains all this and more. All the programs are ready to type in, and include concise instructions on how to enter them and how to use them. With just this one book, you have an instant collection of outstanding Apple software.

> If you prefer to purchase a disk containing all the programs in this book rather than type them in, just use the convenient coupon in the back, or call toll-free 1-800-334-0868.

1

Ideas and Applications

Home Financial Calculator

Patrick Parrish

Though many home budget programs have been published, rarely has there been a program integrating as wide a variety of loan and investment calculations as "Home Financial Calculator." It's versatile, easy to use, and flexible. Rapid recalculation features make it an ideal tool for "what if?" projections. A calculator mode with memory lets you solve problems not directly supported by the program, and you can pass values generated by one calculation to another. It works on all Apple II–series computers, using either DOS 3.3 or ProDOS.

Investment and loan calculations are readily computerized. In fact, many programs have been written which perform these tasks individually. "Home Financial Calculator" goes a step further by integrating several common financial calculations in a menu-driven package. It also features a calculator mode or scratch pad area where program variables can be manipulated using common mathematical operations.

Type in Home Financial Calculator. The program can be entered under either DOS 3.3 or ProDOS, and works on any Apple II–series computer, from the II+ to the IIc. Be careful when typing in the program, especially the long lines which contain the financial formulas. A mistyped program may still run, but its results could be inaccurate. As always, save the program before running it for the first time.

When you run the program, a main menu offers you a choice of Investment or Loan calculations. Type I or L to reach the appropriate submenu.

Common Variables

Before looking at any calculations, let's consider some basics of the program. Home Financial Calculator uses some parameters or variables repeatedly in the calculations. These variables are Total (also referred to as Future Value, Total Owed,

and so on, depending on the calculation); Present Value (principal); Interest Rate; Years; Months; Number of Periods (of either compounding, deposits, withdrawals, or payments, depending on the application); Deposits; and Withdrawals. When in the calculator mode (explained below), you'll reference these eight variables with the single letters T, P, I, Y, M, N, D, and W.

As you work with Home Financial Calculator, the values of the eight variables are preserved until you change them. Whenever the program asks you for an entry (for example, Interest), the current value of that variable is displayed (zero if no value has been entered yet). If you want to keep the current value, just press the Return key. Otherwise, enter the new value and press Return.

With this feature, Home Financial Calculator makes it easy for you to generate "what if?" projections. Simply run the same calculation repeatedly, each time changing a previously entered value. Press Return to keep a value, and change only one or two other values to see the effect on the final result.

You can also store the current value into the calculator mode's Memory Register, or recall a value from the Memory Register. To see how all this works, let's take a look at some calculations possible with Home Financial Calculator.

Investment Calculations

This is what should appear when you type I from the main menu:

```
1) FUTURE VALUE WITH PERIODIC INTEREST
2) FUTURE VALUE WITH INTEREST COMPOUNDED
   CONTINUOUSLY
3) FUTURE VALUE WITH REGULAR DEPOSITS
4) FUTURE VALUE WITH CASH FLOWS
5) WITHDRAWAL OF FUNDS
6) NET PRESENT VALUE
7) CALCULATOR MODE
8) RETURN TO MAIN MENU.
```

Determine which option you want and press the appropriate key.

Each option displays screen prompts which ask you to enter several values. These values are stored in the eight variables already mentioned: T for Total (Future Value), P for Present Value (principal), I for Interest Rate, Y for Years, M for Months, N for Number of Periods, D for Deposits, and W for Withdrawals. Not all calculations require you to enter all these values; others may ask for additional information.

Most calculations can be solved for any *one* of the variables. To solve for a variable, enter an uppercase X at the corresponding prompt. For example, you could enter values for everything *except* the interest rate, typing X at the Interest Rate prompt. Home Financial Calculator then solves for the interest rate.

Remember, however, that the program can solve for only *one* variable during each calculation. If you enter an X at more than one prompt, the program doesn't have enough information to calculate an answer. Keep this in mind, because the program does not check for potential conflicts.

Future Value with Periodic Interest

Home Financial Calculator's options are fairly self-explanatory when you run the program, but let's try an example. We'll calculate the future value of an investment drawing periodic interest. This kind of investment could be a savings account, interest-bearing checking account, bonds, or a money market account. Choose this option by entering 1 at the Investment submenu.

After the screen clears, the program asks for the first input—Future Value, which appears with an asterisk (*). Below this is a zero (the current value of this variable in memory). Following this is a prompt (?).

The asterisk preceding Future Value means that this is one of the variables you can solve for. (A variable *not* preceded by an asterisk means that variable *cannot* be solved for in that particular calculation, so X would be an illegal response.) If you'd like to calculate the Future Value, enter X here, and answer all the other prompts with the appropriate values.

Let's calculate the future value of a $1,000 investment drawing 8 percent interest for two years and three months, with four compounding periods each year. Enter X for *Future Value*, since you'll be solving for this total. Answer *Present*

Value with 1000 (the principal you're investing); *Annual Int Rate (%)* with 8 (enter the percentage, not a fraction); *For # Of Years* with 2; *For # Of Months* with 3; and # *Of Periods (Compounding)* with 4. After you enter the last value, Home Financial Calculator figures the Total Future Value and displays the answer—$1195.09.

Now suppose you wish to know the future value of the same $1,000 investment if you receive 9 percent interest. Choose option 1 on the Investment submenu again and rerun the calculation. Notice how Home Financial Calculator automatically prints the current value of each variable at each prompt. The Future Value prompt shows a current value of 1195.09 from the previous calculation. Type X at this prompt, 9 for Interest Rate, and hit Return at all the other prompts to preserve their values. The result should be $1221.71.

The versatility of Home Financial Calculator becomes apparent when you realize how many different ways you can run this calculation. Using the same menu option, you can calculate the initial investment (or present value) necessary to accrue a certain future value with periodic interest; the interest rate necessary to accrue a future value from a present value; or the time (in years and months) it would take to accumulate a future amount from an initial investment with periodic interest payments. Just enter an X for the unknown value you're seeking, and fill in all the other prompts.

Future Value with Interest Compounded Continuously

Option 2, a variation of option 1, handles investments paying a continuous interest rate. Like option 1, it can handle a number of calculations—just place an X in the slot you'd like to solve for.

Here, after entering all other parameters, you can calculate the future value of an investment; the initial investment required to reach a certain future value; the interest required to reach a desired future value; or the time required to reach a certain future value at a specified interest rate.

Notice that any variables used in option 1 will be displayed with their current values when running option 2. As mentioned above, the eight major variables in Home Financial Calculator retain their values throughout the program until you change them. This feature is convenient when going from one option to another on the Investment or Loan submenu.

In addition, the values are preserved for use in the calculator mode. For instance, you could compare the effect of continuously compounded interest with periodic interest (option 1) without having to retype the input.

Future Value with Regular Deposits

If you're interested in setting up an annuity, you'd choose option 3 on the Investment submenu. You can determine the future value of an account (such as a savings account, Individual Retirement Account [IRA], college or vacation fund, and so on) with regular deposits where interest is compounded with each deposit.

Option 3 can also tell you the amount of each deposit necessary to accrue a future value; the interest rate needed to provide some future value with regular deposits; or the time it would take to amass a future value with regular deposits.

Future Value with Cash Flows

Option 4 does a single calculation—it always solves for Future Value, so don't enter an X anywhere. It calculates the future value of an investment with yearly cash flows (either positive or negative). The Annual Interest Rate you input here is the growth rate on the money you've invested.

As an example, suppose you want to determine the value of a vacation fund collected over four years. You're asked for the number of years, then for the deposit or withdrawal each year. You deposit $500 in the fund the first year and $200 the second. The third year you're forced to withdraw $300 (entered as −300), and the fourth year you put in $400. The fund has a growth rate of 12 percent. Its value after four years will be $1,017.34.

A future value determination can also tell you whether an investment is worthwhile. If the future value of all cash flows is positive or zero, the investment is profitable. A negative future value, on the other hand, represents a losing investment.

Withdrawal of Funds

If you intend to open an account from which you can regularly withdraw funds, choose option 5. With this option, you can determine the initial deposit required in the account to cover your withdrawals; the amount you can withdraw regularly from this account; the rate of interest you must make on

funds in the account; or the period of time over which you can make withdrawals.

Net Present Value

Option 6 lets you determine the feasibility of a prospective investment by calculating its net present value. Net present value is the current value of all future yearly cash flows to an investment along with any initial cash requirement. The interest rate you input here is the rate of return you require on your investment. A positive net present value indicates a profitable investment, while a negative result signifies a losing investment.

Suppose you have the opportunity to make a $2,000 investment which would return $1,500 the first year, cost you $750 the second year, and return $1,900 the third year. You hope to make 13 percent on your money. With option 6, you determine a net present value of $56.87; the investment is therefore profitable.

The Calculator Mode

Option 7 puts you in the calculator mode (also available from the Loan submenu). Calculator mode works very much like a handheld calculator with a single memory. You can type in a value or recall one from a variable by entering its symbol—T(otal), P(resent Value), I(nterest Rate), Y(ears), M(onths), N(umber of Periods), D(eposits), or W(ithdrawals). You can perform simple math on values stored in the Memory Register using reverse Polish notation. And you can use the results in future calculations.

When you enter calculator mode, the calculator command line appears on the screen:

```
V S H R M+ M− M* M/ MR MC MEM=0
```

Here are the commands:

V	View the values of the eight primary variables
S	Store Memory Register into a variable
H	Help—prints the command line
R	Return to main menu, exit calculator mode
M+	Add the last input to the Memory Register
M−	Subtract the last input from the value in the Memory Register, and store the result in the Register
M*	Multiply the last input times the value in the Memory Register, and store the result in the Register
M/	Divide the last input into the value in the Memory Register, and store the result in the Register
MR	Memory Recall
MC	Memory Clear to zero

If you've run through a sample investment calculation, you now have some variables in memory. Enter V in the calculator mode to see them. The screen displays the eight values currently in memory for the eight variables.

To work with one of these variables, enter one of their letters (T, P, I, Y, M, N, D, or W), and press Return. Then type M+ to add it to the Memory Register (all variables must be stored in the Register before you can perform any operations on them). Suppose you put the current value for T into the Register and now wish to add $229 to this value. Enter 229, press Return, then type M+, and press Return. The addition is performed and the result displayed. To store this value back into the T variable, enter S for Store. A prompt appears, requesting the variable in which you intend to store the value. Type T to store the value into the variable T.

You can also use the Memory Register to hold a value not represented by any of the eight variables. To do this, determine a value using the calculator mode and store it into the Memory Register with M+. Then, when you're running a calculation elsewhere in the program, you can substitute this value for any of the eight primary variables by typing MR (Memory Recall) at the appropriate prompt. MR can be used both in the calculator mode and at any prompt where the previous value is displayed.

Finally, option 8 on the Investment submenu returns you to the main menu. Once there, you can perform some loan calculations by typing L.

Loan Calculations

This is what the Loan calculations submenu looks like:

```
1) REGULAR LOAN PAYMENTS
2) REMAINING LOAN LIABILITY
3) FINAL LOAN PAYMENT
4) SINGLE PAYMENT LOAN
5) LOAN AMORTIZATION SCHEDULE
6) CALCULATOR MODE
7) RETURN TO MAIN MENU
```

Regular Loan Payments

Option 1 handles a number of calculations for equal payment loans. You can figure the principal of a loan; the amount of each regular payment necessary to repay a loan; the annual interest rate on a loan with regular payments; or the term of the loan.

Remaining Loan Liability

With option 2, you can determine the remaining balance on a loan with regular payments after a number of payments have been made. Enter the principal on the loan, the amount of each payment, the annual interest rate, the number of payments yearly, and the last payment number.

Final Loan Payment

Option 3 calculates the amount of the final payment on a loan. In many cases, the last payment of a loan will vary from the amount of the regular payment. This option handles situations where the final payment is greater than ("balloon payments") or less than the regular payment.

Single Payment Loan

Option 4 calculates the amount owed on a loan that is paid off with a single payment. You must input the principal on the loan, its annual interest rate, its term in years and months, and the number of times a year the interest on the principal is compounded.

Loan Amortization Schedule

Option 5 displays a loan amortization schedule. Enter the principal on the loan, the amount of each payment, the annual interest rate, the term of the loan, and the number of payments yearly. Then enter the period of the year in which the loan began (for instance, 10 for October) and the range in years of the amortization schedule you'd like to examine.

Because of the complexity of these calculations, there may be a delay before the output appears on the screen, especially if you've chosen to look at the latter years in a long-term loan repayment schedule (such as a home mortgage). When the amortization table appears, it displays the payment number, the beginning balance for the period, the amount paid toward the loan principal, the amount paid in interest, and the ending balance. To keep the information from scrolling off the screen, the program shows only a few payment periods at a time. Press Return to view another screen. When the end of a year is reached, the program gives the total amounts paid on the principal and in interest for the year. In addition, when the last period of the loan is reached, the program displays the final loan payment.

This option is the only one which has a supporting print routine. (The routine itself, in lines 5500–5540, is a slightly revised version of "Apple Screen Dump," another article in this book.) If you want to print out what you see on the screen, simply type P and press Return. Assuming you have a printer connected to your Apple, the amortization schedule is printed and the next screen appears. (The routine works as is with the Apple IIc; you may have to alter lines 5500–5540 slightly if you have an Apple II+ or IIe. Read "Apple Screen Dump" for details.)

The last two options on the Loan submenu are the same as those on the Investment submenu.

Modifying the Program

Home Financial Calculator is written in a modular format for easy modification. For many routines, it uses common input labels (lines 4710–5080) and some output labels (lines 5090–5170). If you want to add an investment or loan calculation routine, choose the labels from these lines that fit your application.

Home Financial Calculator

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
68 100 DIM V(8)
AA 110 V$ = "TPIYMNDW"
D3 120 C$ = "VSHR"
A4 130 C1$ = "M+M-M*M/MRMC"
68 140 Q$ = ""
5F 150 D5 = 13
C8 160 S1 = 5
AC 170 S2 = 15
FD 180 S3 = 23
4F 190 S4 = 31
DE 200 GOSUB 5450
39 210 PRINT "INVESTMENTS AND LOANS"
87 220 PRINT "(I/L) ";
1D 230 INPUT A$
AC 240 IF A$ = "I" THEN 270
7F 250 IF A$ = "L" THEN 2170
1B 260 GOTO 230
EC 270 GOSUB 5450
62 280 PRINT "INVESTMENTS:"
F3 290 PRINT
97 300 PRINT "1) FUTURE VALUE WITH PERIODIC INTEREST
       "
B6 310 PRINT "2) FUTURE VALUE WITH INTEREST COMPOUND
       ED CONTINUOUSLY"
89 320 PRINT "3) FUTURE VALUE WITH REGULAR DEPOSITS"
50 330 PRINT "4) FUTURE VALUE WITH CASH FLOWS"
19 340 PRINT "5) WITHDRAWAL OF FUNDS"
94 350 PRINT "6) NET PRESENT VALUE"
C1 360 PRINT "7) CALCULATOR MODE"
FD 370 PRINT "8) RETURN TO MAIN MENU"
F2 380 PRINT
54 390 PRINT "CHOICE ";
19 400 INPUT A$
82 410 A = VAL (A$)
59 420 IF A < 1 THEN 400
5F 430 IF A > 8 THEN 400
4D 440 ON A GOTO 470,730,970,1360,1550,1940,450,200
EF 450 GOSUB 4180
1A 460 GOTO 200
EE 470 GOSUB 5450
98 480 PRINT "FUTURE VALUE WITH PERIODIC INTEREST"
F5 490 PRINT
D6 500 GOSUB 4710
E8 510 GOSUB 4750
49 520 PRINT "$";
EA 530 GOSUB 4840
```

```
4D 540 PRINT "*";
FE 550 GOSUB 4880
B5 560 IF E = 4 THEN 580
EC 570 GOSUB 4920
03 580 GOSUB 4970
63 590 IF E < > 1 THEN 620
78 600 V(1) = INT (V(2) * (1 + V(3) / V(6)) ^ (V(6)
       * Y) * 100 + .5) / 100
EC 610 GOSUB 5090
18 620 IF E < > 2 THEN 650
11 630 V(2) = INT (V(1) / ((1 + V(3) / V(6)) ^ (V(6)
       * Y)) * 100 + .5) / 100
D8 640 GOSUB 5120
DF 650 IF E < > 3 THEN 680
81 660 V(3) = INT ((V(6) * (V(1) / V(2)) ^ (1 / (V(6
       ) * Y)) - V(6)) * 10000 + .5) / 10000
EA 670 GOSUB 5150
45 680 IF E < > 4 THEN 710
89 690 V(4) = LOG (V(1) / V(2)) / (V(6) * LOG (1 + V
       (3) / V(6)))
E9 700 GOSUB 5180
DB 710 GOSUB 5330
1C 720 GOTO 270
E9 730 GOSUB 5450
5A 740 PRINT "FUTURE VALUE WITH INTEREST COMPOUNDED
       CONTINUOUSLY"
F0 750 PRINT
E4 760 GOSUB 4710
F6 770 GOSUB 4750
57 780 PRINT "*";
F8 790 GOSUB 4840
48 800 PRINT "*";
F9 810 GOSUB 4880
60 820 IF E = 4 THEN 840
E7 830 GOSUB 4920
DD 840 IF E < > 1 THEN 870
31 850 V(1) = INT (V(2) * EXP (V(3) * Y) * 100 + .5)
        / 100
F8 860 GOSUB 5090
43 870 IF E < > 2 THEN 900
1C 880 V(2) = INT (V(1) / EXP (V(3) * Y) * 100 + .5)
        / 100
E4 890 GOSUB 5120
F7 900 IF E < > 3 THEN 930
59 910 V(3) = INT ( LOG (V(1) / V(2)) / Y * 10000 +
       .5) / 10000
E3 920 GOSUB 5150
3E 930 IF E < > 4 THEN 710
C9 940 V(4) = INT ( LOG (V(1) / V(2)) / V(3) * 100 +
       .5) / 100
```

```
F5 950 GOSUB 5180
A2 960 GOTO 710
F3 970 GOSUB 5450
F7 980 PRINT "FUTURE VALUE WITH REGULAR DEPOSITS"
FA 990 PRINT
55 1000 GOSUB 4710
5F 1010 PRINT "*REGULAR DEPOSIT $"
05 1020 C = 6
07 1030 GOSUB 3950
42 1040 PRINT "*";
05 1050 GOSUB 4840
4A 1060 PRINT "*";
AD 1070 GOSUB 4880
09 1080 IF E = 4 THEN 1100
09 1090 GOSUB 4920
0F 1100 GOSUB 4970
F3 1110 IF E < > 1 THEN 1140
7C 1120 V(1) = INT (V(7) * V(6) * ((1 + V(3) / V(6))
        ^ (V(6) * Y) - 1) / V(3) * 100 + .5) / 100
09 1130 GOSUB 5090
08 1140 IF E < > 3 THEN 1280
05 1150 V(3) = .99
F5 1160 I = 0
B9 1170 T = INT (V(7) * (((1 + V(3) / V(6)) ^ (V(6)
      * Y) - 1) / (V(3) / V(6))) * 100 + .5) / 100
4B 1180 TE = ABS (V(3) - I) / 2
E6 1190 I = V(3)
50 1200 IF ABS (T - V(1)) < .005 THEN 1260
57 1210 IF T < V(1) THEN 1240
BF 1220 V(3) = V(3) - TE
78 1230 GOTO 1170
07 1240 V(3) = V(3) + TE
00 1250 GOTO 1170
6B 1260 V(3) = INT (V(3) * 10000 + .5) / 10000
7F 1270 GOSUB 5150
16 1280 IF E < > 4 THEN 1310
E6 1290 V(4) = LOG (V(3) * V(1) / (V(6) * V(7)) + 1)
        / (V(6) * LOG (1 + V(3) / V(6)))
7D 1300 GOSUB 5180
20 1310 IF E < > 7 THEN 710
67 1320 V(7) = INT (V(1) * (V(3) / V(6)) / ((1 + V(3
      ) / V(6)) ^ (V(6) * Y) - 1) * 100 + .5) / 10
      0
83 1330 PRINT
55 1340 PRINT "REGULAR DEPOSITS REQUIRED:$";V(7)
E6 1350 GOTO 710
09 1360 GOSUB 5450
33 1370 PRINT "FUTURE VALUE WITH CASH FLOWS"
97 1380 PRINT
9B 1390 GOSUB 4840
```

```
99 1400 GOSUB 4880
3A 1410 PRINT "CASH FLOW (+/-)"
81 1420 PRINT
52 1430 V(1) = 0
EA 1440 FOR I = 1 TO V(4)
BF 1450 PRINT "CASH FLOW - YEAR #";I
FB 1460 INPUT A$
CE 1470 A = VAL (A$)
B5 1480 V(1) = V(1) + A * (1 + V(3)) ^ (V(4) - I)
97 1490 NEXT I
77 1500 V(1) = INT (V(1) * 100 + .5) / 100
89 1510 GOSUB 5090
81 1520 TE = V(1)
89 1530 GOSUB 5270
E6 1540 GOTO 710
89 1550 GOSUB 5450
95 1560 PRINT "WITHDRAWAL OF FUNDS"
97 1570 PRINT
9F 1580 GOSUB 4750
EA 1590 PRINT "*REGULAR WITHDRAWAL *"
49 1600 C = 7
8B 1610 GOSUB 3950
46 1620 PRINT "*";
89 1630 GOSUB 4840
4E 1640 PRINT "*";
B1 1650 GOSUB 4880
D2 1660 IF E = 4 THEN 1680
8D 1670 GOSUB 4920
B9 1680 GOSUB 4970
21 1690 IF E < > 2 THEN 1720
97 1700 V(2) = INT (V(8) * V(6) / V(3) * (1 - (1 + V
        (3) / V(6)) ^ ( - V(6) * Y)) * 100 + .5) / 1
        00
59 1710 GOSUB 5120
8D 1720 IF E < > 3 THEN 1860
89 1730 V(3) = .99
F9 1740 I = 0
78 1750 R = INT (V(2) * V(3) / V(6) * (1 / ((1 + V(3
        ) / V(6)) ^ (V(6) * Y) - 1) + 1) * 100 + .5)
         / 100
4F 1760 TE = ABS (V(3) - I) / 2
EA 1770 I = V(3)
44 1780 IF ABS (R - V(8)) < .005 THEN 1840
21 1790 IF R < V(8) THEN 1820
C3 1800 V(3) = V(3) - TE
80 1810 GOTO 1750
8B 1820 V(3) = V(3) + TE
88 1830 GOTO 1750
6F 1840 V(3) = INT (V(3) * 10000 + .5) / 10000
83 1850 GOSUB 5150
```

```
A4 1860 IF E < > 4 THEN 1890
52 1870 V(4) = LOG (V(6) * V(8) / (V(6) * V(8) - V(3
     ) * V(2))) / (V(6) * LOG (1 + V(3) / V(6)))
A7 1880 GOSUB 5180
4C 1890 IF E < > 8 THEN 710
21 1900 V(8) = INT (V(2) * V(3) / V(6) * (1 / ((1 +
     V(3) / V(6)) ^ (V(6) * Y) - 1) + 1) * 100 +
     .5) / 100
87 1910 PRINT
EC 1920 PRINT "REGULAR WITHDRAWALS:$";V(8)
EA 1930 GOTO 710
8D 1940 GOSUB 5450
2D 1950 PRINT "NET PRESENT VALUE:$"
9B 1960 PRINT
1E 1970 PRINT "INITIAL INVESTMENT"
ED 1980 C = 1
B1 1990 GOSUB 3950
72 2000 GOSUB 4840
96 2010 GOSUB 4880
37 2020 PRINT "CASH FLOW (+/-)"
7E 2030 PRINT
BA 2040 NV = - V(2)
E7 2050 FOR I = 1 TO V(4)
BC 2060 PRINT "CASH FLOW - YEAR # ";I
F8 2070 INPUT A$
CB 2080 A = VAL (A$)
7C 2090 NV = NV + A / ((V(3) + 1) ^ I)
6E 2100 NEXT I
0E 2110 NV = INT (NV * 100 + .5) / 100
7C 2120 PRINT
61 2130 PRINT "NET PRESENT VALUE:$";NV
22 2140 TE = NV
8A 2150 GOSUB 5270
E7 2160 GOTO 710
8A 2170 GOSUB 5450
76 2180 PRINT "LOANS:"
98 2190 PRINT
8E 2200 PRINT "1) REGULAR LOAN PAYMENTS"
30 2210 PRINT "2) REMAINING LOAN LIABILITY"
97 2220 PRINT "3) FINAL LOAN PAYMENT"
CC 2230 PRINT "4) SINGLE PAYMENT LOAN"
C4 2240 PRINT "5) LOAN AMORTIZATION SCHEDULE"
2C 2250 PRINT "6) CALCULATOR MODE"
A4 2260 PRINT "7) RETURN TO MAIN MENU"
92 2270 PRINT
55 2280 PRINT "CHOICE ";
05 2290 INPUT A$
B1 2300 A = VAL (A$)
D4 2310 IF A < 1 THEN 2290
DF 2320 IF A > 7 THEN 2290
```

```
62 2330 ON A GOTO 2360,2780,2960,3120,3230,2340,200
8C 2340 GOSUB 4180
E0 2350 GOTO 200
8A 2360 GOSUB 5450
89 2370 PRINT "REGULAR LOAN PAYMENTS"
98 2380 PRINT
5D 2390 PRINT "*";
9E 2400 GOSUB 4790
3F 2410 PRINT "*";
4C 2420 GOSUB 5010
47 2430 PRINT "*";
8A 2440 GOSUB 4840
4F 2450 PRINT "*";
B2 2460 GOSUB 4880
F3 2470 IF E = 4 THEN 2490
8E 2480 GOSUB 4920
BA 2490 GOSUB 4970
3E 2500 IF E < > 2 THEN 2550
18 2510 V(2) = INT (V(7) * V(6) / V(3) * (1 - (1 + V
      (3) / V(6)) ^ ( - V(6) * Y)) * 100 + .5) / 1
      00
84 2520 PRINT
62 2530 PRINT "AMT OF PRINCIPAL:$";V(2)
8C 2540 GOTO 2760
D8 2550 IF E < > 3 THEN 2690
92 2560 V(3) = .99
03 2570 I = 0
80 2580 P = INT (V(7) * V(6) / V(3) * (1 - ((1 + V(3
      ) / V(6)) ^ ( - V(6) * Y))) * 100 + .5) / 10
      0
58 2590 TE = ABS (V(3) - I) / 2
CD 2600 I = V(3)
C1 2610 IF ABS (P - V(2)) < .005 THEN 2670
74 2620 IF P < V(2) THEN 2650
8C 2630 V(3) = V(3) + TE
92 2640 GOTO 2580
D4 2650 V(3) = V(3) - TE
9A 2660 GOTO 2580
78 2670 V(3) = INT (V(3) * 10000 + .5) / 10000
8C 2680 GOSUB 5150
66 2690 IF E < > 4 THEN 2720
77 2700 V(4) = - LOG (1 - V(3) * V(2) / (V(6) * V(7)
      )) / (V(6) * LOG (V(3) / V(6) + 1))
8A 2710 GOSUB 5180
56 2720 IF E < > 7 THEN 2760
03 2730 V(7) = INT (V(3) * V(2) / (V(6) * (1 - (V(3)
      / V(6) + 1) ^ ( - V(6) * Y))) * 100 + .5) /
      100
98 2740 PRINT
3E 2750 PRINT "REQ PAYMENT:$";V(7)
```

```
7E 2760 GOSUB 5330
94 2770 GOTO 2170
9A 2780 GOSUB 5450
FF 2790 PRINT "REMAINING LOAN LIABILITY"
82 2800 PRINT
AA 2810 GOSUB 4790
54 2820 GOSUB 5010
8E 2830 GOSUB 4840
AE 2840 GOSUB 4970
79 2850 PRINT "LAST PAYMENT # WAS:"
05 2860 INPUT A$
D7 2870 A = VAL (A$)
1C 2880 FOR J = 1 TO A
A2 2890 I = INT (P * V(3) / V(6) * 100 + .5) / 100
B3 2900 P = P + I - V(7)
83 2910 NEXT J
BA 2920 LI = INT (P * 100 + .5) / 100
90 2930 PRINT
DA 2940 PRINT "LIABILITY AFTER ";A;" PAYMENTS:$";LI
98 2950 GOTO 2760
96 2960 GOSUB 5450
15 2970 PRINT "LAST LOAN PAYMENT"
A4 2980 PRINT
CC 2990 GOSUB 4790
3D 3000 GOSUB 5010
77 3010 GOSUB 4840
65 3020 GOSUB 5050
9B 3030 GOSUB 4970
44 3040 FOR J = 1 TO V(6) * Y
83 3050 I = INT (P * V(3) / V(6) * 100 + .5) / 100
BA 3060 P = P + I - V(7)
8A 3070 NEXT J
F6 3080 LP = INT (P * 100 + .5) / 100 + V(7)
97 3090 PRINT
A3 3100 PRINT "LAST PAYMENT:$";LP
79 3110 GOTO 2760
77 3120 GOSUB 5450
FF 3130 PRINT "SINGLE PAYMENT LOAN"
05 3140 PRINT
AD 3150 GOSUB 4790
8D 3160 GOSUB 4840
7B 3170 GOSUB 5050
B1 3180 GOSUB 4970
33 3190 V(1) = INT (V(2) * (1 + V(3) / V(6)) ^ (Y *
      V(6)) * 100 + .5) / 100
77 3200 PRINT
2B 3210 PRINT "TOTAL OWED:$";V(1)
7F 3220 GOTO 2760
28 3230 C5 = 0
DC 3240 N5 = 0
```

```
C5 3250 F = Ø
84 3260 P1 = Ø
18 3270 I1 = Ø
91 3280 GOSUB 5450
16 3290 PRINT "LOAN AMORTIZATION SCHEDULE"
79 3300 PRINT
A1 3310 GOSUB 4790
4B 3320 GOSUB 5010
85 3330 GOSUB 4840
73 3340 GOSUB 5050
B2 3350 PRINT "# OF PAYMENTS YEARLY"
9B 3360 GOSUB 3950
97 3370 PRINT "ENTER THE PERIOD OF THE YEAR IN WHICH
      THE LOAN BEGAN"
8B 3380 INPUT N
84 3390 NE = N
79 3400 NP = (V(4) * 12 + V(7)) / (12 / V(6))
B7 3410 NY = INT (((N - 1) + NP) / V(6) + .99)
A9 3420 PRINT "ENTER THE RANGE OF YEARS YOU'D LIKE T
     O EXAMINE (FIRST, LAST)"
98 3430 INPUT F1,L1
F5 3440 IF L1 < = NY THEN 3460
2D 3450 L1 = NY
9B 3460 FOR J1 = 1 TO L1
F7 3470 IF J1 < F1 THEN 3490
B1 3480 GOSUB 5390
89 3490 FOR J = 1 TO V(6) - N + 1
79 3500 I = INT (P * V(3) / V(6) * 100 + .5) / 100
B9 3510 N5 = N5 + 1
3A 3520 PP = V(7) - I
7B 3530 IF J1 < > NY THEN 3570
3A 3540 IF N5 < > NP THEN 3570
7A 3550 PP = P
10 3560 F = 1
E9 3570 IF J1 < F1 THEN 3600
87 3580 PRINT N5; TAB( S1); INT (P * 100 + .5) / 100
     ;
8C 3590 PRINT TAB( S2); INT (PP * 100 + .5) / 100;Q$
     ; TAB( S3);
AE 3600 P = P + I - V(7)
3B 3610 IF F = Ø THEN 3640
62 3620 P = Ø
ED 3630 J = V(6)
ED 3640 IF J1 < F1 THEN 3670
E4 3650 PRINT I; TAB( S4); INT (P * 100 + .5) / 100;
97 3660 PRINT
DC 3670 I1 = I1 + I
77 3680 P1 = P1 + PP
A5 3690 C5 = C5 + 1
6E 3700 IF C5 < > D5 THEN 3770
```

```
E4 3710 IF J1 < F1 THEN 3770
6F 3720 GOSUB 5330
87 3730 GOSUB 5450
36 3740 C5 = 0
BB 3750 IF J = V(6) - N + 1 THEN 3770
AF 3760 GOSUB 5390
98 3770 NEXT J
06 3780 IF J1 < F1 THEN 3890
DC 3790 IF F = 0 THEN 3820
83 3800 PRINT
EB 3810 PRINT "FINAL PAYMENT :$"; INT ((PP + I) * 10
      0 + .5) / 100
8B 3820 PRINT
D3 3830 PRINT "TOTAL INT PAID IN YR ";J1;":$"; INT (
      I1 * 100 + .5) / 100
04 3840 PRINT "TOTAL PRINC PAID IN YR ";J1;":$"; INT
       (P1 * 100 + .5) / 100
90 3850 IF F = 1 THEN 3930
02 3860 IF J1 = L1 THEN 3930
85 3870 GOSUB 5330
9D 3880 GOSUB 5450
4C 3890 C5 = 0
7A 3900 P1 = 0
0E 3910 I1 = 0
08 3920 N = 1
EF 3930 NEXT J1
95 3940 GOTO 2760
EF 3950 C = C + 1
27 3960 IF C < > 3 THEN 3990
4E 3970 PRINT V(3) * 100,
81 3980 GOTO 4000
48 3990 PRINT V(C),
72 4000 A$ = ""
E2 4010 INPUT A$
26 4020 IF A$ < > "" THEN 4040
E0 4030 RETURN
D9 4040 IF A$ < > "MR" THEN 4100
71 4050 PRINT "MEM=";M;"  USE AS VARIABLE HERE (Y/N)
     "
F6 4060 INPUT A$
A3 4070 IF A$ = "N" THEN 4000
20 4080 V(C) = M
F8 4090 RETURN
A5 4100 IF A$ < > "X" THEN 4130
69 4110 E = C
DE 4120 RETURN
4E 4130 V(C) = VAL (A$)
49 4140 IF C < > 3 THEN 4160
EE 4150 V(C) = V(C) / 100
EE 4160 RETURN
```

```
23 4170 REM CALCULATOR MODE
90 4180 GOSUB 5450
DF 4190 M5 = 0
64 4200 GOSUB 4530
E6 4210 INPUT A$
92 4220 IF ASC (A$) > 57 THEN 4250
EE 4230 T = VAL (A$)
6C 4240 GOTO 4210
F4 4250 FOR I = 1 TO 8
DA 4260 IF A$ < > MID$ (V$,I,1) THEN 4290
CD 4270 PRINT V(I)
AE 4280 T = V(I)
96 4290 NEXT I
43 4300 FOR J = 1 TO 6
39 4310 IF A$ < > MID$ (C1$,(J - 1) * 2 + 1,2) THEN
      4330
76 4320 ON J GOSUB 4580,4600,4620,4640,4660,4680
81 4330 NEXT J
B3 4340 FOR K = 1 TO 4
3F 4350 IF A$ < > MID$ (C$,K,1) THEN 4370
C8 4360 ON K GOSUB 4410,4460,4530,4560
92 4370 NEXT K
D0 4380 IF M5 = 0 THEN 4210
E3 4390 M5 = 0
DC 4400 RETURN
E8 4410 FOR I = 1 TO 8
56 4420 PRINT MID$ (V$,I,1);"  ";V(I)
82 4430 NEXT I
8C 4440 PRINT
F0 4450 RETURN
81 4460 PRINT "IN WHAT VARIABLE ";
03 4470 INPUT A$
05 4480 FOR I = 1 TO 8
82 4490 IF A$ < > MID$ (V$,I,1) THEN 4510
8B 4500 V(I) = M
7C 4510 NEXT I
E6 4520 RETURN
0A 4530 PRINT C$;" ";C1$;" MEM=";M
8E 4540 PRINT
F2 4550 RETURN
5C 4560 M5 = 1
FA 4570 RETURN
3B 4580 M = M + T
AE 4590 GOTO 4690
1E 4600 M = M - T
90 4610 GOTO 4690
A4 4620 M = M * T
98 4630 GOTO 4690
2F 4640 M = M / T
A0 4650 GOTO 4690
```

```
FA 4660 T = M
A8 4670 GOTO 4690
4B 4680 M = 0
38 4690 PRINT "MEM=";M
E2 4700 RETURN
F4 4710 PRINT "*FUTURE VALUE $"
94 4720 C = 0
98 4730 GOSUB 3950
F2 4740 RETURN
9A 4750 PRINT "*PRESENT VALUE $"
E4 4760 C = 1
A8 4770 GOSUB 3950
03 4780 RETURN
E8 4790 PRINT "PRINCIPAL $"
CE 4800 C = 1
92 4810 GOSUB 3950
5C 4820 P = V(C)
F0 4830 RETURN
27 4840 PRINT "ANNUAL INT RATE (%)"
23 4850 C = 2
A6 4860 GOSUB 3950
01 4870 RETURN
11 4880 PRINT "FOR # OF YEARS"
73 4890 C = 3
90 4900 GOSUB 3950
EA 4910 RETURN
65 4920 PRINT "FOR # OF MONTHS"
9D 4930 C = 4
A0 4940 GOSUB 3950
1E 4950 Y = V(C - 1) + V(C) / 12
FE 4960 RETURN
84 4970 PRINT "# OF PERIODS (COMPOUNDING, DEPOSITS,
      WITHDRAWALS, PAYMENTS) YEARLY"
F1 4980 C = 5
B4 4990 GOSUB 3950
D5 5000 RETURN
64 5010 PRINT "PAYMENTS $"
09 5020 C = 6
0B 5030 GOSUB 3950
E5 5040 RETURN
78 5050 PRINT "TERM OF LOAN:"
AD 5060 GOSUB 4880
05 5070 GOSUB 4920
F5 5080 RETURN
99 5090 PRINT
C4 5100 PRINT "FUTURE VALUE:$";V(1)
DB 5110 RETURN
7F 5120 PRINT
AA 5130 PRINT "REQUIRED INVESTMENT:$";V(2)
E7 5140 RETURN
```

```
8B 5150 PRINT
05 5160 PRINT "ANNUAL INT RATE (%) REQUIRED:";V(3) *
        100
F3 5170 RETURN
D5 5180 V(5) = V(4) - INT (V(4))
1C 5190 V(5) = INT ( INT (12 * V(5) * 10 + .5) / 10)
02 5200 V(4) = INT (V(4))
15 5210 IF V(5) < > 12 THEN 5240
20 5220 V(4) = V(4) + 1
53 5230 V(5) = 0
89 5240 PRINT
AA 5250 PRINT "# OF YEARS AND MONTHS:";V(4);",";V(5)
F1 5260 RETURN
95 5270 PRINT
62 5280 IF TE > = 0 THEN 5310
45 5290 PRINT "THIS IS A LOSING INVESTMENT."
DB 5300 RETURN
69 5310 PRINT "THIS IS A PROFITABLE INVESTMENT."
E3 5320 RETURN
87 5330 PRINT
D6 5340 PRINT "HIT <RETURN> TO CONTINUE"
8D 5350 A$ = ""
FD 5360 INPUT A$
E1 5365 IF A$ = "P" THEN 5500
50 5370 IF A$ < > "" THEN 5360
FB 5380 RETURN
99 5390 GOSUB 5450
8D 5400 PRINT "LOAN AMORTIZATION SCHEDULE FOR YR ";J
      1
86 5410 PRINT "PRIN $";V(2);"  RATE ";V(3) * 100;"%"
      ;"  PAYM $";V(7)
85 5420 PRINT
4C 5430 PRINT "#    BEG BAL    PRINC    INT      END BAL
      "
ED 5440 RETURN
58 5450 HOME
F5 5460 RETURN
DD 5500 D$ = CHR$ (4):I$ = CHR$ (9)
A4 5505 PRINT D$"PR#1"
21 5510 PRINT I$"80N";
59 5515 FOR G7 = 0 TO 2: FOR L7 = 1 TO 8: PRINT SPC(
        20): FOR P7 = 0 TO 39
D5 5520 C7 = PEEK (896 + G7 * 40 + L7 * 128 + P7)
89 5525 PRINT CHR$ (C7);: NEXT : PRINT : NEXT : NEXT
08 5530 PRINT D$"PR#0"
EF 5540 RETURN
```

QUICK.BUT.DUMB
A Simple Terminal Program

Tim Victor

Most modem packages include terminal programs, but just to show you how short one can be, here's "QUICK.BUT.DUMB," a simple terminal routine which lets you access information services like CompuServe and The Source, or any number of electronic bulletin boards. Modem necessary.

If you have a modem connected to your Apple computer, you have the ability to retrieve information from a wide variety of sources. Commercial databases, like CompuServe, Delphi, The Source, and Dow Jones News/Retrieval, provide information on everything from airline schedules to stock quotations. Electronic bulletin board systems (BBSs) let you communicate with other computer owners around the country.

But to do all this, you have to have a terminal program. Many modems include terminal software in their packages, but not all. And many commercial terminal programs, though sophisticated, can be hard to figure out, especially when you're just beginning telecommunications.

Quick

"QUICK.BUT.DUMB" includes a section of machine language to make it fast. The intialization, however, is in BASIC.

Type in and save the program listed below. It uses the default values for the Apple IIc/Apple Modem 300 connection, which are 300 bits per second, seven-bit data, one stop bit, no parity, with full duplex (don't worry if these terms are unfamiliar—you don't have to know anything about telecommunications to use QUICK.BUT.DUMB).

For the Apple II+/IIe, install Super Serial Card II in slot 2, and connect the card for your Apple modem. You also need to change the PR#3 in lines 250 and 340 to PR#0, unless you have an 80-column card in slot 3. Apple II+ owners should also change the last number in line 415 from 0 to 224.

Dumb

Once you have your computer, modem, and telephone line properly connected (refer to your modem's manual for these instructions), load and run QUICK.BUT.DUMB. You're ready to communicate with a variety of information services and bulletin boards.

Of course, QUICK.BUT.DUMB doesn't have download or upload features, a buffer, or an editing system. If you want these features, you will be able to find programs that do almost anything you could want in the way of telecomputing.

The main attraction of this program is its speed in loading, and the speed with which it takes in and puts out information. It's just as accurate as commercial terminal programs.

In some cases, you may even prefer QUICK.BUT.DUMB to more sophisticated programs that take longer to load. If you want to just log on and chat on a BBS, for instance, QUICK.BUT.DUMB is probably the fastest way to get there.

Where's What

Here's a list of some important lines in QUICK.BUT.DUMB, and what they do:

Line(s)	Function
110–120	These lines send a code to the modem that tells it to hang up the phone and disconnect any connection if there is one.
130	This line POKEs in the machine language routine.
250	Sets the IIc screen to 80-column mode.
260–270	Take in the phone number.
300	Sets up input from the modem.
310	Sets up output to the modem.
330	Sends the codes for dialing. AT indicates that a new command string follows, T indicates tone dialing, D tells the modem to dial and enter the originate code, and X$ is the phone number.
340–350	Set output back to the screen and print READY.
390	Calls the machine language loop that handles the serial input/output.

QUICK.BUT.DUMB

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
75 10 REM CHANGE THE LAST DATA IN 415 FROM 0 TO 224
      TO USE THIS PROGRAM ON THE APPLE II+
64 100 PRINT "INITIALIZING...."
4A 110 PRINT CHR$ (4)"PR#2"
8C 120 PRINT "+++";
3C 130 FOR I = 768 TO I + 43: READ A: POKE I,A: NEXT
E9 240 PRINT
31 250 PRINT CHR$ (4)"PR#3
3A 260 PRINT "PHONE NUMBER TO DIAL";
3C 270 INPUT X$
3B 300 PRINT CHR$ (4)"IN#2"
4C 310 PRINT CHR$ (4)"PR#2"
E6 320 PRINT
32 330 PRINT "ATTD"X$
3# 340 PRINT CHR$ (4)"PR#3
8A 350 PRINT "READY"
F4 390 CALL 768
5B 400 DATA 173,169,192,41,8
C4 410 DATA 240,21,173,168,192,9,128
DC 415 DATA 201,225,144,6,201,251,176,2,105,0
C# 420 DATA 32,240,253,76,0,3,173,0,192
FE 430 DATA 16,223,141,16,192,41,127
AC 440 DATA 141,168,192,76,0,3
```

Softball Statistics

Roger Felton
Translation by Patrick Parrish

When it's time to get ready for your softball league, you'll want to keep track of all the individual and team results. With "Softball Statistics," it's easy. You can enter data for each player's times at bat, hits, runs, and so on. The program automatically computes batting averages, stores cumulative results on disk as the season progresses, generates formatted printouts with sorted rankings for all players, and more. For all Apple II–series computers, using either DOS 3.3 or ProDOS. (An 80-column printer is optional, but recommended.)

What's the worst position on a softball team? Catchers have to squat in an uncomfortable stance for an hour or more and duck hazardous foul balls. Pitchers have to duel with mighty sluggers and dodge powerful line drives. The players on the bases have to stretch their bodies like rubber bands to nab wayward throws from their teammates while keeping at least one toe on the bag. And outfielders have to scoop up bouncing grounders, knowing that no one is backing them up besides the outfield fence.

But as demanding as all these positions are, there's another that could be worse—that of team statistician. Keeping track of your teammates' performances is often a laborious, thankless job. Sometimes the statistician is a reserve player or friend of the team who doesn't even get to play. Caged in the dugout, the statistician is supposed to document every hit, run, and walk, and boost team morale by contributing lively chatter. After the game, the statistician has to spend hours punching numbers into a calculator to figure out the batting averages.

"Softball Statistics" makes that job much easier. After each game, the program prompts you to enter vital stats for each player. Then it automatically calculates the batting averages and prints sorted rankings on the screen or printer. It can also print sorted rankings for hits, runs, and runs batted in. These game statistics can then be merged with data for all previous games, and updated season results can be sorted by

category and printed. Finally, the program lets you store the cumulative statistics on disk.

If you're a professional baseball or Little League fan, you can use Softball Statistics to follow the fortunes of your favorite team. And with modifications, the program could be adapted to a wide variety of sports.

Typing the Program

Entering Softball Statistics is easy when you use the "Apple Automatic Proofreader," an error-checking program you'll find in Appendix B. Once you have a copy of Apple Proofreader typed in and saved on disk, load and run it before you start entering Softball Statistics.

Entering the Roster

The next step is to enter your team's roster into the program. Softball Statistics can handle a team with up to 20 players, and it stores this information in DATA statements as part of the program itself.

The DATA statements begin at line 2070 and must conform to a predefined format: a two-digit jersey number followed by a space, then the player's first or last name. Precede one-digit jersey numbers with a zero, for example, 08 for 8. Names can be any length, but only the first seven characters appear on the printouts. Each entry is separated by a comma. Here's an example:

```
2070 DATA 23 LEE,17 JACKSON,33 JOHNSTON,10
     LONGSTREET,04 HILL
```

In the output, JOHNSTON and LONGSTREET would appear as JOHNSTO and LONGSTR.

The program is listed with dummy entries in the DATA statements, such as 44 JIM and 11 PLAYERX. Substitute your own team members for these entries. If your team has fewer than 20 players, leave the remaining dummy entries in the DATA statements; the program must have 20 entries to function, and it ignores the PLAYERX entries.

Finally, put your own team's name in the string statement at line 140. With these adjustments, Softball Statistics is now ready to run.

Important note: You should avoid tinkering with the player name DATA statements once you've started using the program. Otherwise, there will be problems when it attempts to compute cumulative season totals. If you drop a player from the roster and replace him or her with another player, the new player's totals will contain the old player's results as well. To drop a player, substitute a PLAYERX dummy entry at that position in the DATA statement. Of course, this means the dropped player's results will no longer be included in the team totals for the season. If you wish to retain a dropped player's results in the team totals, leave the player's name in the DATA statement and enter 999 in response to all input prompts following subsequent games (see below).

Compiling Statistics

Once the roster is entered, you can run the program. It begins by asking for individual game statistics. The first prompt reads:

WHO DID YOU PLAY?

Respond with the opposing team's name—such as KELLY'S—and press Return. The next prompt asks:

ENTER YOUR SCORE AND THEIR SCORE (SEPARATED BY A COMMA):

For instance, if your team lost by a score of 9 to 5, you'd type 5,9 and press Return.

The program now begins asking for individual player statistics. If the first player name on your roster is DARRYL, the program prints

DARRYL'S STATISTICS FOR THIS GAME:

and then prompts you, one by one, to enter the number of times at bat, runs scored, hits, runs batted in (RBIs), doubles, triples, home runs, and walks. At each prompt, type the appropriate number and press Return. After the last prompt, the program continues to the next player on the roster and repeats the cycle.

If a player missed a game, type 999 at the first prompt. This automatically enters zeros for all his or her stats and skips to the next player. In fact, entering 999 at any prompt inputs zeros for all of a player's remaining game stats.

Individual Printouts

After you type the last statistic for the last player, the program prints the message WORKING while it sorts all the data. (The WORKING message appears at other points in the program during sorts, since the sort routine is written in BASIC and isn't particularly fast.) In a few moments, the program displays

DO YOU WANT A PRINTOUT OF THE GAME'S STATS (Y/N)?

Type Y for yes or N for no. If you press N, the program asks if you want to input data for another game. If you press Y, it asks

TO SCREEN OR PRINTER (S/P)?

Type S or P. Softball Statistics then prints the individual stats for all team members for that game, sorted in descending order by batting averages. Figure 1 shows a printed example from a recent COMPUTE! softball game.

Figure 1. Game Stats

```
STATISTICS FOR THIS GAME:
COMPUTE! VS HOMESITTERS SECURITY    SCORE:5-21

ROSTER IS SORTED BY BATTING AVERAGE

#  PLAYER     AB   RUNS   HITS   RBI   2B   3B   HR   BB    AVG
55 STOKES      2      1      2     0    0    0    0    0  1.000
08 KNICK       2      0      1     0    0    0    0    0  0.500
03 SMITH       2      1      1     0    1    0    0    0  0.500
06 BATEMAN     2      1      1     1    0    0    0    0  0.500
11 KRUSENT     2      1      1     0    0    0    0    0  0.500
10 KEIZER      2      1      1     0    0    0    0    0  0.500
08 BOWMAN      2      0      1     1    0    0    0    0  0.500
44 POST        3      0      1     2    0    1    0    0  0.333
09 MYKYTYN     2      0      0     0    0    0    0    0  0.000
07 MILLER      2      0      0     1    0    0    0    0  0.000
22 KRAUSE      3      0      0     0    0    0    0    0  0.000

TOTALS        24      5      9     5    1    1    0    0  0.375
```

Because the output is formatted for an 80-column printer, it looks odd—but is still readable—on screens with less than 80 columns. By pressing any key, you can stop the screen or printer output at any time. Start output again by pressing P.

Next, the program requests:

DO YOU WANT SORTED PRINTOUTS OF HIT, RBI, AND RUN LEADERS (Y/N)?

Again, type Y for yes or N for no. If you type N, the program asks if you want to input stats for another game. If you answer Y, it asks again if you want the output directed to the screen or printer, and then prints sorted rankings for the various slugging categories for that game. As before, you can stop the output by pressing any key and restart it by pressing P.

Figure 2. Team Leaders

HITS SORT:

#	PLAYER	HITS
55	STOKES	2
06	BATEMAN	1
08	BOWMAN	1
11	KRUSENT	1
10	KEIZER	1
03	SMITH	1
08	KNICK	1
44	POST	1
09	MYKYTYN	0
07	MILLER	0
22	KRAUSE	0
TOTAL HITS		9

RBIS SORT:

#	PLAYER	RBIS
44	POST	2
08	BOWMAN	1
07	MILLER	1
06	BATEMAN	1
11	KRUSENT	0
10	KEIZER	0
09	MYKYTYN	0
08	KNICK	0
55	STOKES	0
03	SMITH	0
22	KRAUSE	0
TOTAL RBIS		5

RUNS SORT:

#	PLAYER	RUNS
10	KEIZER	1
06	BATEMAN	1
55	STOKES	1
03	SMITH	1
11	KRUSENT	1
09	MYKYTYN	0
08	KNICK	0
07	MILLER	0
08	BOWMAN	0
22	KRAUSE	0
44	POST	0
TOTAL RUNS		5

Finally, the program prompts you:

DO YOU WANT TO INPUT STATS FROM ANOTHER GAME (Y/N)?

Usually, you type N at this prompt, unless you're entering results of more than one game. If you type Y, the program repeats the entire process described above.

Season Totals

Softball Statistics makes it easy for you to tabulate running totals for the entire season by storing game results on disk. After you've entered and viewed the stats for the most recent game, the program asks

WOULD YOU LIKE TO MERGE IN DATA FOR THE YEAR (Y/N)?

The first time you run Softball Statistics, of course, you won't have any previous data on disk, so you'd answer N, skipping to the next prompt. During subsequent runs, you'd

answer Y to merge in data for the year. The program then requests a filename for the disk file and merges these existing stats with the results you've entered for the latest game or games.

Season totals are then computed automatically, and the program shows:

DO YOU WANT A PRINTOUT OF THE YEAR'S STATS (Y/N)?

If you type N, you're asked to specify a filename to save the updated data file, and the program ends. If you answer Y, the program asks if you want output directed to the screen or printer, and it then prints season totals for all players. This printout includes the team's win/loss record and sorts players in descending order by batting averages.

Figure 3. Year's Stats

```
STATISTICS FOR THE YEAR:
RECORD FOR THE YEAR:  WINS:0 LOSSES:1

ROSTER IS SORTED BY BATTING AVERAGE

#  PLAYER     AB   RUNS   HITS   RBI   2B   3B   HR   BB    AVG
55 STOKES      2      1      2     0    0    0    0    0  1.000
08 KNICK       2      0      1     0    0    0    0    0  0.500
03 SMITH       2      1      1     0    1    0    0    0  0.500
06 BATEMAN     2      1      1     1    0    0    0    0  0.500
11 KRUSENT     2      1      1     0    0    0    0    0  0.500
10 KEIZER      2      1      1     0    0    0    0    0  0.500
08 BOWMAN      2      0      1     1    0    0    0    0  0.500
44 POST        3      0      1     2    0    1    0    0  0.333
09 MYKYTYN     2      0      0     0    0    0    0    0  0.000
07 MILLER      2      0      0     1    0    0    0    0  0.000
22 KRAUSE      3      0      0     0    0    0    0    0  0.000

TOTALS        24      5      9     5    1    1    0    0  0.375
```

Afterward, the program asks if you want sorted printouts for hits, RBIs, and runs—again, based on season totals (these charts resemble those in Figure 2). Finally, the program gives you the opportunity to save the updated data file on disk until the next game.

Softball Computing

If you're interested in programming, you can learn a lot by studying Softball Statistics because it's written entirely in BASIC. In fact, the input/output routine beginning at line

3000 and the printing routine starting at line 4000 are general enough to be adapted to your own programs.

You don't have to be a programmer, though, to appreciate Softball Statistics. If you're a softball statistician, no longer do you have the worst position on the team. Maybe it's the shortstop....

Softball Statistics

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
38 90 D$ = CHR$ (4):I$ = CHR$ (9)
3C 100 D5 = 4
8D 110 D6 = 1
E9 120 PL = 20
7B 130 DIM B(9),CC(20),IN(21),ST(8),RT(20,8),TT(20,8
   ),F$(8),NA$(20),R$(21)
AA 140 TM$ = "TEAM NAME"
EA 150 C$ = "0000"
21 160 FOR I = 1 TO 8
8E 170 READ F$(I)
ED 180 NEXT I
F4 190 FOR J = 1 TO PL
39 200 READ NA$(J)
5D 210 NA$(J) = MID$ (NA$(J),1,10)
63 220 NEXT J
E9 230 FOR J = 1 TO PL
6B 240 R$(J) = MID$ (NA$(J),1, LEN (NA$(J))) + MID$
   (" ",1,10 - LEN (NA$(J)))
20 250 FOR I = 1 TO 8
8E 260 TT(J,I) = 0
4C 270 ST(I) = 0
EE 280 NEXT I
71 290 NEXT J
CC 300 GOSUB 2030
33 310 PRINT "GAME STATISTICS"
A3 320 PRINT "WHO DID YOU PLAY"
1F 330 INPUT OT$
F3 340 PRINT "ENTER YOUR SCORE AND THEIR SCORE (SEPA
   RATED BY A COMMA)"
2F 350 INPUT YS,TS
FE 360 W = W + ABS (YS > TS)
60 370 L = L + ABS (TS > YS)
F4 380 FOR J = 1 TO PL
53 390 IF MID$ (NA$(J),4,7) < > "PLAYERX" THEN 420
65 400 R$(J) = R$(J) + "0000000000000000000000000000
   00000.000"
12 410 GOTO 600
```

```
D1 420 GOSUB 2030
8F 430 PRINT MID$ (NA$(J),4, LEN (NA$(J)));"'S STATI
       STICS FOR THIS GAME:"
20 440 FOR I = 1 TO 8
B6 450 B(I) = 0
8B 460 PRINT F$(I)
A4 470 INPUT B(I)
97 480 IF LEN ( STR$ (B(I))) > = D5 THEN 450
50 490 IF B(I) < > 999 THEN 540
B1 500 FOR K = I TO 8
EF 510 B(K) = 0
E6 520 NEXT K
21 530 I = 8
E9 540 NEXT I
E5 550 GOSUB 1350
25 560 FOR I = 1 TO 8
10 570 RT(J,I) = RT(J,I) + B(I)
32 580 TT(J,I) = TT(J,I) + B(I)
F3 590 NEXT I
63 600 NEXT J
EC 610 GOSUB 1670
0E 620 MM = 0
20 630 FOR I = 1 TO 8
EF 640 FOR J = 1 TO PL
DA 650 ST(I) = ST(I) + TT(J,I)
6F 660 NEXT J
15 670 B(I) = ST(I)
F2 680 NEXT I
F7 690 R$(J) = ""
DD 700 GOSUB 1350
AD 710 TT$ = R$(J)
E9 720 GOSUB 1560
B9 730 PRINT "DO YOU WANT TO INPUT STATS FROM ANOTHE
       R GAME (Y/N)?"
E5 740 GOSUB 1920
CF 750 IF A$ = "Y" THEN 230
DC 760 GOSUB 2030
9F 770 PRINT "WOULD YOU LIKE TO MERGE IN DATA FOR TH
       E YEAR (Y/N)?"
ED 780 GOSUB 1920
C5 790 IF A$ = "N" THEN 840
0D 800 C = 1
CC 810 GOSUB 3010
99 820 W = SW + W
75 830 L = SL + L
F4 840 GOSUB 1670
F3 850 FOR J = 1 TO PL
28 860 FOR I = 1 TO 8
DA 870 IF A$ = "N" OR MID$ (NA$(J),4,7) = "PLAYERX"
       THEN 920
```

```
1C 880 B(I) = VAL ( MID$ (R$(J),11 + (I - 1) * 4,4))
54 890 B(I) = RT(J,I) + B(I)
38 900 RT(J,I) = B(I)
9B 910 GOTO 930
E8 920 B(I) = RT(J,I)
4B 930 ST(I) = 0
ED 940 NEXT I
DB 950 R$(J) = MID$ (R$(J),1,10)
EB 960 GOSUB 1350
74 970 NEXT J
5D 980 MM = 1
2F 990 FOR I = 1 TO 8
75 1000 FOR J = 1 TO PL
49 1010 ST(I) = ST(I) + RT(J,I)
74 1020 NEXT J
BF 1030 B(I) = ST(I)
7B 1040 NEXT I
85 1050 R$(J) = ""
77 1060 GOSUB 1350
17 1070 TT$ = R$(J)
65 1080 GOSUB 2030
FF 1090 PRINT "DO YOU WANT A PRINTOUT OF THE YEAR'S
      STATS (Y/N)?"
61 1100 GOSUB 1920
99 1110 IF A$ = "N" THEN 1140
85 1120 GOSUB 1670
7D 1130 GOSUB 1560
28 1140 PRINT "DO YOU WANT TO SAVE THE DATA (Y/N)?"
75 1150 GOSUB 1920
E9 1160 IF A$ = "Y" THEN 1180
E0 1170 END
1E 1180 C = 2
76 1190 GOTO 3010
E6 1200 REM SHELL SORT
7D 1210 FOR J = 1 TO PL
BD 1220 IN(J) = J
45 1230 CC(J) = VAL ( MID$ (R$(J),BB,E))
80 1240 NEXT J
A2 1250 FOR J = PL - 1 TO 1 STEP - 1
17 1260 FOR I = 1 TO J
A0 1270 IF CC(IN(I)) > CC(IN(I + 1)) THEN 1310
6E 1280 TE = IN(I)
FC 1290 IN(I) = IN(I + 1)
90 1300 IN(I + 1) = TE
75 1310 NEXT I
7A 1320 NEXT J
E3 1330 RETURN
B0 1340 REM BUILD R$
32 1350 IF B(1) = 0 THEN 1380
38 1360 IF B(3) = 0 THEN 1380
```

```
78 1370 GOTO 1410
25 1380 B(9) = 0
92 1390 AV$ = "0.000"
62 1400 GOTO 1420
CD 1410 B(9) = INT (B(3) / B(1) * 1000 + .5) / 1000
      + .0001
E9 1420 FOR I = 1 TO 8
0E 1430 B$ = STR$ (B(I))
42 1440 B$ = MID$ (C$,1,D5 - LEN (B$)) + MID$ (B$,D6
      , LEN (B$))
0E 1450 R$(J) = R$(J) + B$
0B 1460 NEXT I
34 1470 IF B(9) = 0 THEN 1530
57 1480 AV$ = STR$ (B(9))
E4 1490 IF MID$ (AV$,1,1) < > " " THEN 1510
DD 1500 AV$ = MID$ (AV$,2,6)
E0 1510 IF MID$ (AV$,1,1) < > "." THEN 1530
78 1520 AV$ = "0" + AV$
EB 1530 R$(J) = R$(J) + MID$ (AV$,1,5)
EB 1540 RETURN
EA 1550 REM SORT BY AVERAGES
7F 1560 BB = 43
03 1570 E = 5
63 1580 GOSUB 1210
27 1590 IF MM = 1 THEN 1630
51 1600 GOSUB 2030
47 1610 PRINT "DO YOU WANT A PRINTOUT OF THE GAME'S
     STATS (Y/N)?"
73 1620 GOSUB 1920
BD 1630 IF A$ = "N" THEN 1660
9B 1640 GOSUB 1960
71 1650 GOTO 4010
F5 1660 RETURN
99 1670 PRINT
C0 1680 PRINT "WORKING..."
02 1690 RETURN
7F 1700 PRINT
05 1710 PRINT "DO YOU WANT SORTED PRINTOUTS OF HIT,
     RBI, AND RUN LEADERS (Y/N)?"
75 1720 GOSUB 1920
C1 1730 IF A$ = "N" THEN 1760
9D 1740 GOSUB 1960
96 1750 GOTO 1770
F7 1760 RETURN
A5 1770 GOSUB 1670
10 1780 BB = 19
CE 1790 E = 4
4B 1800 GOSUB 1210
B0 1810 I = 3
4B 1820 GOSUB 5000
```

```
78 1830 BB = 23
5B 1840 GOSUB 1210
01 1850 I = 4
5B 1860 GOSUB 5000
0A 1870 BB = 15
6B 1880 GOSUB 1210
90 1890 I = 2
45 1900 GOSUB 5000
E7 1910 RETURN
65 1920 GET A$
99 1930 IF A$ = "" THEN 1920
6A 1940 IF (A$ < > "Y") AND (A$ < > "N") THEN 1920
F7 1950 RETURN
9B 1960 PRINT
B0 1970 PRINT "TO SCREEN OR PRINTER (S/P)?"
05 1980 GET P$
81 1990 IF P$ = "" THEN 1980
C9 2000 IF (P$ < > "P") AND (P$ < > "S") THEN 1980
4B 2010 DE = (P$ = "S")
DA 2020 RETURN
45 2030 HOME
E2 2040 RETURN
12 2050 DATA TIMES AT BAT,RUNS,HITS,RBIS,DOUBLES,TRI
     PLES,HOME RUNS,WALKS
72 2060 REM LIST PLAYERS BY NUMBER & NAME
C6 2070 DATA 44 JIM,22 PETE,03 JOHN,08 KEN,55 MIKE
5E 2080 DATA 06 BARRY,07 BILL,08 BOB,09 MARTY,22 LOU
     IE
61 2090 DATA 11 PLAYERX,12 PLAYERX,13 PLAYERX,14 PLA
     YERX,15 PLAYERX
8B 2100 DATA 16 PLAYERX,17 PLAYERX,18 PLAYERX,19 PLA
     YERX,20 PLAYERX
71 3000 REM INPUT/OUTPUT ROUTINE
3E 3010 HOME
95 3020 PRINT "ENTER DATA FILE NAME:": INPUT FF$: ON
     ERR GOTO 3100
16 3030 PRINT D$;"OPEN ";FF$: IF C = 2 THEN 3070
DB 3040 PRINT D$;"READ ";FF$: INPUT SW,SL: FOR J = 1
     TO PL: INPUT R$(J)
4D 3050 R$(J) = MID$ (NA$(J),1, LEN (NA$(J))) + MID$
     (" ",1,10 - LEN (NA$(J))) + R$(J)
40 3060 NEXT J: GOTO 3080
9D 3070 PRINT D$;"WRITE ";FF$: PRINT W; CHR$ (13);L:
     FOR J = 1 TO PL: PRINT MID$ (R$(J),11,32):
     NEXT J
E3 3080 PRINT D$;"CLOSE ";FF$: POKE 216,0: IF C = 2
     THEN END
F7 3090 RETURN
D1 3100 HOME : VTAB 5: PRINT "ERROR # "; PEEK (222);
     " OCCURRED AT LINE "; PEEK (219) * 256 + PEE
     K (218)
```

```
4C 3110 VTAB 10: PRINT "HINT: HAVE YOU PREVIOUSLY SA
     VED THE": PRINT "DATA FILE TO DISK?"
C5 3120 PRINT D$;"CLOSE ";FF$: PRINT : PRINT "TRY AG
     AIN.": PRINT : PRINT "HIT ANY KEY": GET A$
26 3130 IF C = 1 THEN 810
64 3140 GOTO 3010
ED 4000 REM PRINT ROUTINE
33 4010 PRINT : IF DE = 1 THEN 4030
EB 4020 PRINT D$;"PR#1": PRINT I$;"80N"
3F 4030 IF MM = 1 THEN T$ = "THE YEAR": GOTO 4050
EA 4040 T$ = "THIS GAME"
D6 4050 PRINT "STATISTICS FOR "T$":": IF MM = 1 THEN
     4070
65 4060 PRINT TM$" VS "OT$"     SCORE:"YS"-"TS: GOTO
     4080
3A 4070 PRINT "RECORD FOR THE YEAR:  WINS:"W" LOSSES
     :"L
36 4080 PRINT : PRINT "ROSTER IS SORTED BY BATTING A
     VERAGE": PRINT
07 4090 PRINT "#  PLAYER     AB    RUNS    HITS     RBI
        2B      3B      HR      BB      AVG"
6C 4100 FOR J = 1 TO PL: IF MID$ (R$(IN(J)),4,7) = "
     PLAYERX" THEN 4160
03 4110 PRINT MID$ (R$(IN(J)),1,10)" ";: FOR I = 1 T
     O 8:Q = 0: FOR K = 0 TO 3: IF MID$ (R$(IN(J)
     ),11 + (I - 1) * 4 + K,1) < > "0" THEN Q = 1
E7 4120 IF MID$ (R$(IN(J)),11 + (I - 1) * 4 + K,1) =
     "0" AND Q = 0 AND K = 3 THEN PRINT "0";: GO
     TO 4150
5A 4130 IF MID$ (R$(IN(J)),11 + (I - 1) * 4 + K,1) =
     "0" AND Q = 0 THEN PRINT " ";: GOTO 4150
3C 4140 PRINT MID$ (R$(IN(J)),11 + (I - 1) * 4 + K,1
     );
03 4150 NEXT K: PRINT "   ";: GOSUB 5120: NEXT I: PR
     INT "  " MID$ (R$(IN(J)),43,5)
80 4160 NEXT J: PRINT : PRINT "TOTALS      ";
D2 4170 FOR I = 1 TO 8:Q = 0: FOR K = 1 TO 4: IF MID
     $ (TT$,(I - 1) * 4 + K,1) < > "0" THEN Q = 1
D2 4180 IF MID$ (TT$,(I - 1) * 4 + K,1) = "0" AND Q
     = 0 AND K = 4 THEN PRINT "0";: GOTO 4210
48 4190 IF MID$ (TT$,(I - 1) * 4 + K,1) = "0" AND Q
     = 0 THEN PRINT " ";: GOTO 4210
FF 4200 PRINT MID$ (TT$,(I - 1) * 4 + K,1);
68 4210 NEXT K: PRINT "   ";: NEXT I: PRINT "  " MID
     $ (TT$,33,5)
E4 4220 PRINT : IF DE = 0 THEN PRINT D$;"PR#0"
6B 4230 GOTO 1700
FF 5000 PRINT : IF DE = 1 THEN 5020
E8 5010 PRINT D$;"PR#1": PRINT I$;"80N"
```

```
1D 5020 PRINT :T = 0: PRINT : PRINT F$(I)" SORT:": P
    RINT
A1 5030 PRINT "#  PLAYER      "F$(I): FOR J = 1 TO PL
    : IF MID$ (R$(IN(J)),4,7) = "PLAYERX" THEN 5
    090
39 5040 PRINT MID$ (R$(IN(J)),1,10)"     ";:Q = 0: FO
    R K = 0 TO 3: IF MID$ (R$(IN(J)),BB + K,1) <
     > "0" THEN Q = 1
CF 5050 IF MID$ (R$(IN(J)),BB + K,1) = "0" AND Q = 0
    AND K = 3 THEN PRINT "0": GOTO 5080
DD 5060 IF MID$ (R$(IN(J)),BB + K,1) = "0" AND Q = 0
    THEN PRINT " ";: GOTO 5080
6F 5070 PRINT MID$ (R$(IN(J)),BB + K,1);: IF K = 3 T
    HEN PRINT ""
2B 5080 NEXT K:T = T + VAL ( MID$ (R$(IN(J)),BB,E)):
    GOSUB 5120
29 5090 NEXT J: PRINT : PRINT "TOTAL ";F$(I);"      "
    ;T: PRINT
1D 5100 IF DE = 0 THEN PRINT D$;"PR#0"
DB 5110 RETURN
3C 5120 A = PEEK ( - 16384): IF A < 128 THEN 5150
FF 5130 A = PEEK ( - 16384): IF A < 128 THEN POKE -
    16368,0: GOTO 5130
CB 5140 A$ = CHR$ (A - 128): IF A$ < > "P" THEN 5130
EB 5150 RETURN
```

Apple Screen Dump

Donald W. Watson

Thirty seconds is all it takes to get a printer dump of your Apple II text screen. With dozens of uses, this simple subroutine can easily be inserted into your own programs. For all Apple II–series computers (with slight modifications) using either DOS 3.3 or ProDOS.

If you own a computer, you probably have a printer, too. Although the electronic revolution may have promised a paperless office, it just didn't happen. Having a piece of paper in your hands, with the information you need on it, is just too important. You can't take your computer *everywhere;* it won't fold up and go in your pocket.

"Apple Screen Dump" lets you quickly get a printed copy of whatever text or figures happen to be on your monitor or television screen. In fact, one program in this book, "Home Financial Calculator," uses the screen dump routine to print out information.

Just Seven Lines

Boot your system into Applesoft BASIC, type NEW to clear the program memory, type HOME to clear the screen, and then enter the following seven short lines:

```
100 D$=CHR$(4):I$=CHR$(9)
105 PRINT D$"PR#1"
110 PRINT I$"80N";
115 FOR G=0 TO 2:FOR L=1 TO 8:PRINT SPC(20):FOR P=0
    TO 39
120 C=PEEK(896 + G * 40 + L * 128 + P)
150 PRINT CHR$(C);:NEXT:PRINT:NEXT:NEXT
160 PRINT D$"PR#0"
```

If you have an Apple II, II+, or IIe computer with a parallel printer interface card in slot 1, you can use the program exactly as shown; if you have a serial printer interface card, delete the second statement in line 100 (:I$=CHR$(9)), and

eliminate line 110 completely. If you have an Apple IIc, you can use the routine as is.

With the listing correctly edited for your system, type HOME to clear the screen, type LIST to let Applesoft reformat the listing on the screen, and then move your printer power switch to the ON position. If you wanted to print a copy of the listing, you would ordinarily have to set up the printer with (at least) an immediate mode PR#1 command followed by an immediate mode LIST command. Since the program you just entered is a screen dump program, why not use it to print itself? Just enter RUN and watch your system dump the full 960-character screen from the Apple II text screen memory to your printer.

The Okidata Microline 80 Parallel Printer will dump the text screen memory in about one and a half minutes. The Qume Sprint 5/55 Serial Printer will dump it in one minute. Both Apple's ImageWriter and Scribe printers will print out the full screen in less than 30 seconds.

Screen Organization

Lines 100–110 are explained in the printer control card manuals. Line 110 is required in the program if a parallel printer interface control card is used; in addition to setting the printer to accept 80-character lines, it directs output to the printer only—"freezing" the screen display while screen memory is dumped.

Line 115 sets up three loop functions, indexing the dump routine to the requirements of the Apple II text screen memory address plan. Refer to your Apple computer's reference manual for a map of the text screen. The screen is organized into three vertical sections, or groups (G = 0 TO 2), of eight lines each (L = 1 TO 8). Each line contains 40 addresses for the characters to be printed (P = 0 TO 39). The PRINT SPC(20) statement provides a 20-character left-hand margin to center the printed record in an 80-character horizontal print format. If you want printing to be left-justified, just delete the PRINT SPC(20) statement from this line.

At line 120, the three loop indices from line 115 are used with an offset starting value (896) in an expression to yield each successive text screen memory address. The expression yields the first screen position, decimal address (1024) for G = 0, L = 1, and P = 0; and it yields the correct value for each

of the remaining 959 memory addresses as the loop variables are incremented. The PEEK function returns the decimal value for the contents of each text screen memory address, and the line finally assigns that value to variable C.

Line 150 directs the printer to print the ASCII character identified by the decimal value of the variable C, and terminates each of the index loops at the appropriate increments. The PRINT statement provides a linefeed and carriage return for each group of 40 characters printed.

The gap in the program line numbers is significant. The program, as you typed it in, will dump the text screen memory correctly only if the memory does not contain INVERSE or FLASH mode character codes. Insert the following three lines to convert INVERSE and FLASH character codes to NORMAL mode character codes:

```
130 IF C < 32 THEN C=C + 192
135 IF C > 31 AND C < 96 THEN C=C + 128
140 IF C > 95 AND C < 128 THEN C=C + 64
```

Using It as a Subroutine

The program is easily converted to a subroutine for use in other Applesoft II programs. Just add a line 170 with a RETURN statement and call the subroutine from your program code with a line containing a GOSUB 100 statement. Of course, you can change the line numbers of the routine and put it anywhere you want.

That's what was done in "Home Financial Calculator," a program listed elsewhere in this book. Take a look at lines 5500–5540 in that program, and you'll find a slightly modified version of the screen dump routine. Line 5365 instructs the program to go to the routine.

```
5365 IF A$="P" THEN 5500
```

The figure below is one of the amortization schedule screens from Home Financial Calculator. HIT <RETURN> TO CONTINUE normally lets the user move to the next screen at his or her own pace. The program code supporting the prompts contains an IF-THEN statement (line 5365) to detect the character P. No visual prompt is needed, but typing P and hitting Return calls the text screen dump subroutine.

With line 5365 present, a P response at any of the loan amortization schedule screens dumps that screen to the printer, producing a hardcopy, as illustrated below.

Printed Copy of the Screen

```
LOAN AMORTIZATION SCHEDULE FOR YR 2
PRIN $60000  RATE 11.5%  PAYM $645

#    BEG BAL    PRINC    INT      END BAL
3    59859.33   71.35    573.65   59787.98
4    59787.98   72.03    572.97   59715.95
5    59715.95   72.72    572.28   59643.23
6    59643.23   73.42    571.58   59569.81
7    59569.81   74.12    570.88   59495.69
8    59495.69   74.83    570.17   59420.86
9    59420.86   75.55    569.45   59345.31
10   59345.31   76.27    568.73   59269.04
11   59269.04   77.01    567.99   59192.03
12   59192.03   77.74    567.26   59114.29
13   59114.29   78.49    566.51   59035.8
14   59035.8    79.24    565.76   58956.56

TOTAL INT PAID IN YR 2:$6837.23
TOTAL PRINC PAID IN YR 2:$902.77

HIT <RETURN> TO CONTINUE
?P
```

When you add this screen dump routine to your own program, however, keep one thing in mind. Before your program enters the subroutine, make sure that a carriage return has been executed. You can do this with a simple PRINT statement.

Potential Uses

Screen dump copy is especially useful in inventory management systems. While filling orders, a stock clerk can refer to the computer inventory files to get a screen display of what's on hand and the bin location for a part number. The screen dump copy can be carried to the bin, the parts picked to fill the order, and the dump copy (marked with the quantity

picked) becomes the transaction record for later use in correcting the computer inventory file information.

With a little screen format and label format planning, a text screen dump can be used to print labels or envelopes for addresses selected from a mailing or shipping file.

There are many more very practical uses, of course. The benefits of the text screen dump routine are that it's short, simple, and accessible—you can modify it to suit your own application requirements. Subroutines specifically written to format a report directly to the printer can often be avoided by use of the text screen dump.

An Even Shorter Method

If DOS is not present, if the screen contains no INVERSE or FLASH mode characters, *and* if you're using a serial interface control card, the following one-line program (about 70 bytes) will dump the Apple II text screen to your printer. You can also use this one-line routine to dump Apple IIc screens.

```
100 PR$1:FOR G=0 TO 2:FOR L=1 TO 8:FOR P=0 TO 39:C=
    PEEK(896 + G * 40 + L * 128 + P):PRINT CHR$(C);
    :NEXT:PRINT:NEXT:NEXT:PR#0
```

Fast Filer

Richard Mansfield and Patrick Parrish

Maintain a master index of magazine articles with this short BASIC program for all Apple II–series computers, using either DOS 3.3 or ProDOS.

How many times have you been working on a program when you recall a magazine article that has just the information you need—but finding it is another matter? That is, you know the article's *somewhere* in the house—but where? You could spend hours paging through back issues to find what you're looking for. Now, with "Fast Filer," you'll have a fast and easy way to retrieve such information.

Searching the Database

Fast Filer is designed for simplicity and convenience. To search the database, all you really need to do is type RUN and follow the prompts. The program first asks whether you want to send output to the screen or to the printer. Then the menu displays several options. You can search the database in several different ways: by magazine title, by author, by subject, by publication date, or by two categories at once.

For example, say, you want to list all articles from *COMPUTE!* magazine. Simply choose option 1 and enter COMPUTE! when prompted for the magazine name. To list all articles by Charles Brannon, choose option 2 and enter BRANNON in response to the author prompt. Once the listing begins, you can pause it by pressing any key, and resume by pressing P.

Fast Filer accepts abbreviations, so it's usually not necessary to type in the entire name. You can abbreviate COMPUTE! as COMPU, for example. However, you must give Fast Filer enough information to distinguish similar names. If the database contains articles by Butterfield and Buncombe, entering BU for the author lists all articles by *both* authors, since both names share those two characters. Entering BUT would distinguish the two names and list all Butterfield articles.

For added flexibility, options 5 and 6 let you search by more than one category at a time. Option 5 provides an *AND*

function to find articles that *share* two categories: For instance, to find all *COMPUTE!* articles written by Charles Brannon, select option 5 and enter 1,2 (be sure to separate the numbers with a comma). Then enter the magazine and author names as prompted.

Option 6 provides an *OR* function to find articles in *either* of two categories. Perhaps you're interested in machine language. With option 6 you could find every article that was categorized under the subject MACHINE LANGUAGE, *or* that was written by Jim Butterfield (who often writes on that subject). The ability to search two categories simultaneously is very powerful. Just as with the AND search feature, after selecting this option, enter the two menu item numbers, separated by a comma.

Easy Data Entry

Of course, no database is useful until it contains some data. Line 1999 of Fast Filer is a template that shows the format for entering data. To enter new data, simply add new lines to Fast Filer, using line numbers higher than 1999. (Lines 2000–2040 are examples which you can modify or delete.)

Every new entry must be in the form of a BASIC line consisting of a line number and a DATA statement, followed by six data items separated by commas. Here's the format:

MAGAZINE TITLE, AUTHOR, SUBJECT, DATE, PAGE NUMBER, COMMENTS

Because Fast Filer separates data items with commas, you must not put commas within the data itself. For instance, enter BRANNON C for an author's name, *not* BRANNON, C.

You cannot omit any of the data items for a particular entry; if you do, the entire list of data is thrown out of sequence. Instead of leaving a particular item blank, substitute something like N/A (for not applicable). For example, you might have an entry for which you don't feel the need to add a comment, like

2000 DATA COMPUTE!,READERS FEED-BACK,ILLOGICAL APPLE LOGIC,6/85,12,N/A

Pay particular attention to line 2050, which tells Fast Filer it has reached the end of the data. This must *always* be the last DATA line in the program. (It doesn't have to be numbered 2050; it just has to be the last DATA line of the pro-

gram.) When adding new data, renumber this line accordingly. When you've finished adding data, resave Fast Filer on disk or tape. The next time you run it, all the new data will be available. Since the data is appended to the program itself, the size of the database is limited only by your computer's memory.

Designing a Database

Fast Filer provides the basic framework for a database, but for maximum flexibility, it leaves the most important design choices up to you. You're free to choose whatever subject categories you like, making them as general or as specific as your needs require.

Creating categories deserves some careful forethought. Clearly, a subject category like COMPUTERS is too broad to be useful. On the other hand, the subject must have enough breadth to encompass more than one article. Consistency is essential, too. If you pick MACHINE LANGUAGE as a subject, then stick with that subject name; categorizing other machine language articles under subject names like ML or MACH LANG will result in incomplete searches.

Before adding your first entry, you may want to decide on standard names for your major categories. These could be saved for future reference in a written list or added to Fast Filer as REM statements.

Use consistent names for magazine titles and authors as well. If you enter a magazine title as COMPUTE (without the exclamation point), it won't be found when you search for articles under the keyword COMPUTE! (although the reverse would work). Likewise, APPLE AP is a more convenient title than COMPUTE!'S APPLE APPLICATIONS.

Fast Filer's ability to abbreviate can work to your advantage. For instance, say that you pick GRAPHICS as a major category. If you enter graphics articles under subject names like GRAPHICS II+, GRAPHICS IIE, GRAPHICS IIC, and so on, then Fast Filer can find *all* graphics articles (under the subject GRAPHICS) as well as graphics articles for a particular computer.

There are limits to what Fast Filer can do, of course, as there are with any BASIC program this brief. But its simplicity makes the program easier to customize. One of the best ways to learn programming is to begin with an existing program and alter it to fit your own needs. Such changes can range

from the purely cosmetic to more significant improvements. In fact, with only a few modifications, Fast Filer can be used to index practically anything, from books or record albums to investments, rare coins, or stamps.

Fast Filer

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
56 10 D$ = CHR$ (4):I$ = CHR$ (9): DIM A$(6):G$ = "
                "
9F 20 HOME : VTAB 10: HTAB 4: PRINT "PRINT TO SCREEN
      OR PRINTER (S/P)?"
C3 30 A = PEEK ( - 16384): IF A < 128 THEN 30
8A 40 K$ = CHR$ (A - 128): POKE - 16368,0: IF K$ < >
      "P" AND K$ < > "S" THEN 30
7A 50 DE = (K$ = "S"):LABEL$(1) = "MAGAZINE TITLE:":
      LABEL$(2) = "AUTHOR'S LAST NAME:"
6D 60 LABEL$(3) = "THE TARGET SUBJECT:":LABEL$(4) =
      "DATE (IE., 1/14/85 OR 1/85):"
43 70 HOME : VTAB 6: PRINT G$"CHOOSE ONE (1-8):": PR
      INT : PRINT G$" 1. MAGAZINE"
6B 80 PRINT G$" 2. AUTHOR": PRINT G$" 3. SUBJECT": P
      RINT G$" 4. DATE"
FF 90 PRINT G$" 5. AND": PRINT G$" 6. OR": PRINT G$"
      7. PRINT ALL": PRINT G$" 8. QUIT"
4B 100 A = PEEK ( - 16384): IF A < 128 THEN 100
B0 110 K$ = CHR$ (A - 128): POKE - 16368,0: IF VAL (
      K$) < 1 OR VAL (K$) > 8 THEN 100
07 120 K = VAL (K$): ON K GOTO 130,140,150,160,170,1
      70,310,360
86 130 C = 1: GOTO 370
A8 140 C = 2: GOTO 370
CA 150 C = 3: GOTO 370
EC 160 C = 4: GOTO 370
9B 170 H$ = "OR": IF K = 5 THEN H$ = "AND"
AA 180 PRINT : PRINT G$"# "H$" # (1-4):": PRINT G$;:
       INPUT N1,N2
E8 190 IF (N1 < 1 OR N1 > 4) OR (N2 < 1 OR N2 > 4) T
      HEN 180
92 200 HOME : PRINT "TYPE "LABEL$(N1): INPUT I1$:L =
       LEN (I1$)
37 210 PRINT : PRINT "TYPE "LABEL$(N2): INPUT I2$:L2
       = LEN (I2$)
8E 220 PRINT :Q = 0:F = 0: RESTORE : IF DE = 0 THEN
      PRINT D$;"PR#1": PRINT I$;"80N"
A3 230 GOSUB 480: IF F = 1 THEN 430
20 240 IF K = 6 THEN 270
```

```
57 250 IF LEFT$ (A$(N1),L) < > I1$ OR LEFT$ (A$(N2),
       L2) < > I2$ THEN 290
2Ø 260 GOTO 28Ø
E5 270 IF LEFT$ (A$(N1),L) < > I1$ AND LEFT$ (A$(N2)
       ,L2) < > I2$ THEN 29Ø
9C 280 Q = 1: GOSUB 5ØØ
Ø6 290 IF F = Ø THEN 23Ø
11 300 GOTO 43Ø
B6 310 HOME :F = Ø: RESTORE : IF DE = Ø THEN PRINT D
       $;"PR#1": PRINT I$;"8ØN"
C2 320 GOSUB 48Ø: IF F = 1 THEN 34Ø
34 330 GOSUB 5ØØ: IF F = Ø THEN 32Ø
BD 340 IF DE = Ø THEN PRINT D$;"PR#Ø"
1D 350 GOTO 45Ø
97 360 END
Ø7 370 HOME : PRINT "TYPE "LABEL$(C): INPUT INP$:L =
        LEN (INP$)
9B 380 PRINT :Q = Ø:F = Ø: RESTORE : IF DE = Ø THEN
       PRINT D$;"PR#1": PRINT I$;"8ØN"
BØ 390 GOSUB 48Ø: IF F = 1 THEN 43Ø
32 400 IF LEFT$ (A$(C),L) < > INP$ THEN 42Ø
9Ø 410 Q = 1: GOSUB 5ØØ
CA 420 IF F = Ø THEN 39Ø
BC 430 IF DE = Ø THEN PRINT D$;"PR#Ø"
Ø5 440 IF Q = Ø THEN PRINT : PRINT G$;: INVERSE : PR
       INT "NO MATCHES FOUND": NORMAL
15 450 PRINT : PRINT G$;: INVERSE : PRINT "PRESS ANY
        KEY": NORMAL : POKE - 16368,Ø
3C 460 A = PEEK ( - 16384): IF A < 128 THEN 46Ø
45 470 POKE - 16368,Ø: GOTO 7Ø
7B 480 READ A$(1),A$(2),A$(3),A$(4),A$(5),A$(6): IF
       A$(1) = "END" THEN F = 1
26 490 RETURN
5E 500 PRINT A$(1)"   "A$(2)"   "A$(3)"   "A$(4)"
       P. "A$(5)"   "A$(6): PRINT
83 510 A = PEEK ( - 16384): IF A < 128 THEN RETURN
54 520 A = PEEK ( - 16384): IF A < 128 THEN 52Ø
54 530 A$ = CHR$ (A - 128): POKE - 16368,Ø: IF A$ <
       > "P" THEN 52Ø
1D 540 RETURN
41 1999 REM DATA TEMPLATE: MAGAZINE,AUTHOR,SUBJECT,D
       ATE,PAGE,COMMENTS
F8 2000 DATA COMPUTE!,SCHULTZ N,MINDBUSTERS,4/85,44,
       GAME
23 2010 DATA GAZETTE,BRANNON C,HORIZONS,1/85,8Ø,VIC
       TO 64
54 2020 DATA GAZETTE,RANDALL N,ROAD TO MOSCOW,12/84,
       8Ø,GAME REVIEW
```

```
87 2030 DATA COMPUTE!,WATSON D,APPLE SCREEN DUMP,10/
        84,169,TEXT SCREEN
AD 2040 DATA COMPUTE!,KEES M,SUPERBASIC 64,10/83,198
        ,ADDS 35 COMMANDS TO BASIC
47 2050 DATA END,0,0,0,0,0
```

2

Games

Heat Seeker

Tim Victor
(Original game concept by Jeff Wolverton)

Your jet climbs upward to avoid the missile, then dives for the ground. But it's still on your tail. You can't shake a heat seeker. A fast-action, machine language game for Apple II–series computers using DOS 3.3 or ProDOS. Joystick or keyboard control.

Heat-seeking missiles are dangerous. They sense the heat from your jet engine and home in on you. They'll catch you, too—they're faster than a jet.

Your assignment: Eliminate the heat-seeker base. It's easy enough to strafe the missiles on the ground, but if any are launched, you'll have to take evasive action.

AppleMLX

"Heat Seeker" is written entirely in machine language (ML). Only with ML are such impressive graphics and speed possible. Usually, ML programs are difficult to enter. The mass of numbers which make up an ML program can seem endless, and typing them all in without a single error is almost impossible. That's why we've listed Heat Seeker in MLX format. You need to use "AppleMLX," a machine language entry program found in Appendix C, to type in Heat Seeker. Before you begin entering the game, then, you'll need a copy of AppleMLX on disk.

Once you have a copy of MLX, load and run it. Two prompts appear on the screen, asking for the starting and ending address of the program you're going to type in. For Heat Seeker, those addresses are

```
START ADDRESS? 1000
END ADDRESS?   2377
```

Just follow the instructions outlined in the article which accompanies AppleMLX. You can type in Heat Seeker in as many sessions as you want, saving your work on disk and returning to it later.

ProDOS Modification. If you're typing in Heat Seeker under the ProDOS operating system, you need to change one

line of the program listing. Use the following line instead of what appears in the listing.

```
1788: 10 AD 51 C0 4C 00 BE 20 60
```

When you've finished entering Heat Seeker, you'll have a file on disk, ready to run. If your Apple can display 80 columns of text, you *must* have it in 40-column mode before running the game. Press the Escape key, then Control-Q to enter 40-column mode.

Now type

BRUN ***filename*** (where *filename* is the program's name as it shows on your disk)

and hit Return. The game loads and runs, and you'll see the options menu screen. Either one or two people can play Heat Seeker. The two-player mode isn't competitive—players take turns flying the plane, trying for the highest score. You need only one joystick, even with two players.

There are three levels of difficulty: Trainer, Novice, and Ace. All you need to do is press the T, N, or A key. In the Trainer level, missiles are never launched. You can fly as much as you like without being chased, but crashes are counted against your eight-jet total. This is the perfect level to become familiar with the controls of your aircraft. Novice and Ace are actual playing levels: Ace has faster action and tighter curves. All three levels award a flight-time bonus of ten points every few seconds, just for staying in the air.

Piloting the Jet

After you've selected your options, the game screen appears. Press the joystick fire button or the space bar to start your jet. Use the joystick or keyboard to control the movement of the plane. The controls may seem a little confusing at first. If you're using the joystick, pull back to loop upward (counterclockwise) and push forward to loop down (clockwise), like a real airplane. The key controls are a bit different. The right-arrow key moves the jet downward and the left-arrow key loops the aircraft upward. The jet moves at a constant velocity—you can't speed up or slow down. Press either the joystick button or the space bar to launch a missile at the heat seekers on the ground. It does no good to fire at a moving heat seeker. They're equipped with an Improved Electronic

Evasion (IEE) circuit which makes them impossible to hit. The only way to get rid of a seeker is to make it crash into the ground. When you're being pursued, dive for the ground and pull up at the last second. Seekers are faster, but they can't turn as quickly.

Your plane can't shoot itself, so don't worry about all your shots zipping across the screen. If you manage to eliminate all the heat seekers, you get to start all over again with a new group of heat seekers. A bonus of 1000 points is awarded for each group you eliminate. You have eight jets to work with—the number remaining is displayed on the screen, next to the score.

After crashing, you can catch your breath for a moment. You'll notice that all eight missiles have been replaced, even though you might have destroyed several with the previous jet. When you're ready to fly again, press the joystick button or the space bar to continue.

At the end of a game (after you've crashed eight jets), you can go back to the options menu to change selections, quit the game, or play the same game configuration again. (If you're using joystick controls, pressing the button gives you another play.)

Speedy Addiction

Heat Seeker is a commercial-quality, arcade-style game. With the exception of sound (omitted because of program length), Heat Seeker has all the elements you expect from something you've paid up to $20 for. It's fast, smooth, and graphically dazzling. More important, it's addicting.

Heat Seeker

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program. If you're using ProDOS, note the one-line change mentioned in the article.

```
START ADDRESS: 1000
END ADDRESS:   2377

1000: 20 E8 1F A0 5F 99 00 01 3B
1008: 88 10 FA A0 08 A9 0A 99 6E
1010: 32 01 88 88 D0 F7 4C 0F D1
1018: 17 2C 50 C0 2C 57 C0 2C 51
1020: 52 C0 2C 54 C0 A9 00 05 96
1028: 1C 0D 40 00 0D 41 00 A9 E5
1030: 20 0D 43 00 05 E6 20 F6 33
```

```
1038: F3 A9 40 8D 42 80 85 E6 A3
1040: 20 F6 F3 20 48 15 A9 00 98
1048: A0 04 99 56 81 99 5B 81 FC
1050: 88 10 F7 8D 99 81 A9 08 BE
1058: 8D 80 81 8D 81 81 A9 01 CE
1060: 8D 31 81 A9 00 8D 6B 1D 88
1068: 8D 85 81 8D 10 C0 A2 0A 8C
1070: 9D 31 81 CA CA D0 F9 8D A3
1078: 3C 81 A9 60 8D 3D 81 A9 60
1080: 20 8D 00 81 A9 28 8D 0C 41
1088: 81 A9 75 8D 18 81 A9 00 75
1090: 8D 24 81 A9 FF A0 00 99 67
1098: 48 81 C8 C0 0E D0 F8 A9 B1
10A0: 80 8D 48 81 8D 54 81 20 66
10A8: C7 12 A9 01 A0 07 99 8F 5A
10B0: 81 88 10 FA 20 E0 1A 20 3E
10B8: 94 12 20 FE 15 AD 42 80 FF
10C0: AC 43 80 8D 43 80 8C 42 68
10C8: 80 C9 40 A9 00 2A AA 9D D9
10D0: 54 C0 20 BE 16 AD 85 81 2F
10D8: F0 F8 A9 00 8D 85 81 A9 14
10E0: 0C 8D 53 81 20 E0 1A A2 48
10E8: 07 BD 8F 81 D0 0C DE 8F 0A
10F0: 81 8E 8E 81 20 C4 14 AE 4A
10F8: 8E 81 CA 10 EC 20 AE 16 76
1100: AD 48 81 D0 03 20 87 14 04
1108: A2 03 BD 49 81 D0 03 20 FD
1110: 31 14 CA 10 F5 AD 4D 81 AC
1118: D0 03 4C 9E 14 AD 4E 81 4C
1120: D0 08 8D 33 81 A9 80 8D D2
1128: 48 81 A2 03 BD 4F 81 D0 52
1130: 08 20 31 14 A9 FF 9D 49 97
1138: 81 CA 10 F0 AD 54 81 C9 6A
1140: FF D0 08 20 D8 15 A9 80 88
1148: 8D 54 81 20 FE 15 20 80 85
1150: 1B A2 00 8E 8B 81 BD 31 A0
1158: 81 D0 03 4C ED 11 20 B6 3F
1160: 13 AD 86 81 C9 BC 90 17 D9
1168: E0 04 90 06 DE 31 81 4C 79
1170: ED 11 A9 54 9D 0C 81 A9 12
1178: 00 9D 0D 81 20 6A 13 E0 6D
1180: 04 90 03 A9 04 2C A9 0A F1
1188: 8D 8C 81 A9 00 8D 8E 81 34
1190: A9 06 8D 8D 81 AD 86 81 E4
1198: C9 AD 90 51 BD 00 81 ED 11
11A0: 8D 81 B0 07 6D 8C 81 90 A1
11A8: 44 B0 04 C9 07 B0 2B AC 34
11B0: 8E 81 B9 8F 81 C9 01 D0 B0
11B8: 34 A9 54 9D 0C 81 A9 00 7D
11C0: 9D 0D 81 20 6A 13 AC 8E AE
```

```
11C8: 81 A9 00 99 BF 81 20 C4 37
11D0: 14 20 BC 15 AE BB 81 4C E0
11D8: ED 11 18 AD BD 81 69 11 6A
11E0: BD BD 81 EE BE 81 AD BE B0
11E8: 81 C9 08 D0 AF E8 E8 E0 20
11F0: 0C F0 03 4C 53 11 AD 4D 02
11F8: 81 C9 FF D0 0B 20 BE 16 C7
1200: AD 85 81 F0 03 20 DF 13 07
1208: AD 33 81 F0 2A AD 4E 81 35
1210: C9 FF D0 23 20 40 1C AD 4D
1218: 44 1C C9 04 90 04 C9 FC 04
1220: 90 15 AD 45 1C C9 04 90 7C
1228: 04 C9 FC 90 0A A2 00 20 64
1230: 6A 13 A2 02 20 6A 13 AD 41
1238: 43 1C 8D 6C 1D 20 68 1D D4
1240: 20 3B 16 20 94 12 AD 42 92
1248: 80 AC 43 80 8D 43 80 8C 4F
1250: 42 80 C9 40 A9 00 2A AA 3F
1258: BD 54 C0 A2 00 A0 0A B9 03
1260: 31 81 F0 01 CA 88 88 10 45
1268: F6 A0 05 B9 4D 81 C9 FF 70
1270: F0 02 CA CA 88 10 F4 18 1A
1278: 8A 69 08 90 14 F0 12 8D 5C
1280: 89 81 A9 F0 8D 8A 81 CE 76
1288: 8A 81 D0 FB CE 89 81 D0 9C
1290: F1 4C E4 10 A2 00 BD 31 20
1298: 81 F0 03 20 A5 12 E8 E8 4C
12A0: E0 0C D0 F2 60 BD 00 81 FC
12A8: 8D 85 19 BD 0C 81 8D 86 FB
12B0: 19 BD 0D 81 8D 87 19 BD 05
12B8: 3C 81 8D 83 19 BD 3D 81 01
12C0: 8D 84 19 20 80 19 60 20 3B
12C8: EB 12 AD 42 80 AC 43 80 FE
12D0: 8D 43 80 8C 42 80 C9 40 4D
12D8: A9 00 2A AA BD 54 C0 20 A2
12E0: EB 12 A9 00 8D 40 80 8D B0
12E8: 41 80 60 20 E0 1A A9 07 A5
12F0: 8D 85 19 A9 56 8D 86 19 0A
12F8: A9 80 8D 87 19 A9 8C 8D 52
1300: 83 19 A9 60 8D 84 19 20 3A
1308: 80 19 18 AD 85 19 69 11 07
1310: 8D 85 19 C9 8C 90 F0 AD 54
1318: 42 80 09 13 85 F9 A9 80 B9
1320: 85 F8 A0 50 A9 2A 91 F8 72
1328: C8 A9 55 91 F8 C8 C0 78 C5
1330: D0 F2 A5 F9 18 69 04 85 C3
1338: F9 29 1C D0 E5 A9 03 85 97
1340: 24 A9 00 20 5B FB AD 42 4D
1348: 80 85 E6 AD 99 81 D0 03 3F
1350: A9 37 2C A9 34 20 03 15 76
```

```
1358: A9 1B 85 24 AD 99 81 D0 B4
1360: 03 A9 34 2C A9 37 20 03 29
1368: 15 60 A9 00 9D 18 81 9D 54
1370: 24 81 A9 54 DD 0C 81 B0 56
1378: 08 9D 0C 81 A9 00 9D 0D 39
1380: 81 A9 E0 9D 3C 81 A9 60 63
1388: 9D 3D 81 8A 4A A8 A9 40 2E
1390: 99 4D 81 C0 00 D0 03 8C E8
1398: 6B 1D C0 01 D0 0E A9 00 F5
13A0: 8D 43 1C 8E 8B 81 20 D4 42
13A8: 15 AE 8B 81 C0 02 90 05 C2
13B0: A9 FF 99 47 81 60 BD 3C 98
13B8: 81 85 F8 BD 3D 81 85 F9 F0
13C0: A0 00 B1 F8 85 FC C8 B1 60
13C8: F8 85 FD 88 BD 0C 81 0A 40
13D0: 18 71 FC 8D 86 81 BD 0D 9A
13D8: 81 30 03 CE 86 81 60 A9 BD
13E0: 00 8D 85 81 A2 04 BD 31 05
13E8: 81 F0 07 E8 E8 E0 0C 90 EE
13F0: F5 60 AD 00 81 69 04 C9 63
13F8: 8C 90 02 E9 8C 9D 00 81 C4
1400: AD 0C 81 69 02 9D 0C 81 E8
1408: AD 18 81 9D 18 81 AD 24 5D
1410: 81 9D 24 81 A9 EE 9D 3C 7D
1418: 81 A9 60 9D 3D 81 FE 31 70
1420: 81 8A 4A E9 01 AA A9 30 C9
1428: 9D 49 81 A9 0C 8D 53 81 FA
1430: 60 8A 48 0A 69 04 AA A9 2F
1438: 00 9D 31 81 68 AA 60 AD 62
1440: 98 81 F0 06 A9 80 8D 48 46
1448: 81 60 DE 8F 81 8E 8E 81 03
1450: 20 C4 14 EE 33 81 AD 8E B4
1458: 81 8D 02 81 0A 0A 0A 0A 93
1460: 6D 02 81 69 07 8D 02 81 7A
1468: A9 54 8D 0E 81 A9 00 8D 4D
1470: 0F 81 A9 8B 8D 26 81 A9 20
1478: 00 8D 1A 81 A9 08 8D 3E 26
1480: 81 A9 60 8D 3F 81 60 A2 1C
1488: 07 BD 8F 81 C9 01 F0 AF 91
1490: CA 10 F6 AD 98 81 D0 03 4B
1498: 20 B8 15 4C 5E 10 AD 99 8E
14A0: 81 AA DE 80 81 4D 97 81 0A
14A8: 8D 99 81 A8 B9 80 81 D0 5C
14B0: 0D EC 99 81 F0 0B BD 80 95
14B8: 81 F0 06 8E 99 81 4C 5E 51
14C0: 10 4C 39 17 AD 8E 81 85 CC
14C8: F8 0A 0A 65 F8 69 D1 85 1E
14D0: F8 AD 42 80 09 16 85 F9 D6
14D8: A2 0F A9 00 A8 91 F8 C8 91
14E0: C0 03 D0 F9 20 EB 14 CA 87
```

```
14E8: D0 F0 60 A5 F9 18 69 04 23
14F0: 85 F9 29 1C D0 0C A5 F8 3C
14F8: 69 80 85 F8 A5 F9 69 E0 FE
1500: 85 F9 60 85 45 86 46 84 25
1508: 47 A2 1F 86 1B 0A 0A 0A E9
1510: 18 69 28 85 1A 90 02 E6 FB
1518: 1B A5 28 85 08 A5 29 29 E8
1520: 03 05 E6 85 09 A2 08 A0 C5
1528: 00 B1 1A A4 24 91 08 E6 AA
1530: 1A D0 02 E6 1B A5 09 18 E3
1538: 69 04 85 09 CA D0 E8 A5 6A
1540: 45 A6 46 A4 47 4C F0 FD 15
1548: A9 20 85 E6 20 57 15 A9 A0
1550: 40 85 E6 20 57 15 60 A9 54
1558: 00 85 24 20 5B FB A2 01 7B
1560: A0 14 BD 17 19 20 03 15 6D
1568: BD 19 19 20 03 15 88 D0 2B
1570: F1 CA 10 EC A9 04 85 24 A3
1578: A9 00 20 5B FB A0 42 A2 BA
1580: 13 20 AD 15 AD 97 81 F0 03
1588: 07 A0 2F A2 0E 20 AD 15 CF
1590: A9 04 85 24 A9 01 20 5B 70
1598: FB A0 21 A2 13 20 AD 15 C0
15A0: AD 97 81 F0 07 A2 0E A0 46
15A8: 0E 20 AD 15 60 B9 1A 19 20
15B0: 20 03 15 88 CA D0 F6 60 BE
15B8: A9 01 D0 1E A9 00 20 F4 76
15C0: 15 BD 59 81 18 69 05 C9 62
15C8: 0A B0 04 9D 59 81 60 E9 F9
15D0: 09 9D 59 81 A9 02 D0 02 23
15D8: A9 03 20 F4 15 FE 56 81 BE
15E0: BD 56 81 C9 0A D0 0C A9 A1
15E8: 00 9D 56 81 CA 10 EE E0 B2
15F0: 05 10 EA 60 AE 99 81 F0 D4
15F8: 03 18 69 05 AA 60 A0 04 44
1600: B9 56 81 99 3E 1E 88 10 F3
1608: F7 A9 06 8D 43 1E 20 38 3F
1610: 1E AD 97 81 F0 13 A0 04 DA
1618: B9 5B 81 99 3E 1E 88 10 4D
1620: F7 A9 1E 8D 43 1E 20 38 5A
1628: 1E AE 99 81 BD 80 81 8D DA
1630: 3E 1E A9 13 8D 43 1E 20 3F
1638: 3B 1E 60 AD 4D 81 C9 FF 74
1640: D0 11 AE 18 81 AC 24 81 F8
1648: 20 6F 16 8D 3C 81 A9 60 97
1650: 8D 3D 81 AD 4E 81 C9 FF A9
1658: D0 14 AE 1A 81 AC 26 81 F5
1660: 20 6F 16 18 69 70 8D 3E 23
1668: 81 A9 60 8D 3F 81 60 A9 0F
1670: 00 C0 32 90 04 C0 CE 90 6D
```

```
1678: 08 E0 80 90 23 A9 04 D0 92
1680: 1F C0 80 B0 02 A9 80 E0 20
1688: 32 90 0A E0 CE B0 06 09 8F
1690: 01 E0 80 90 02 09 02 C9 90
1698: 80 90 05 18 49 7F 69 09 6F
16A0: 8D 82 81 18 0A 6D 82 81 72
16A8: 0A 6D 82 81 0A 60 A2 0E C2
16B0: BD 47 81 C9 FF F0 03 DE 03
16B8: 47 81 CA D0 F3 60 AD 88 54
16C0: 81 F0 43 AD 00 C0 10 2D 7D
16C8: A0 06 8C 83 81 2C 10 C0 2E
16D0: C9 88 D0 06 A9 01 8D 6B 56
16D8: 1D 60 C9 95 D0 06 A9 FF 30
16E0: 8D 6B 1D 60 C9 A0 D0 18 E2
16E8: AD 53 81 C9 FF D0 06 A9 86
16F0: 01 8D 85 81 60 AD 83 81 0C
16F8: F0 05 CE 83 81 F0 01 60 23
1700: A9 00 8D 6B 1D 60 AD 61 92
1708: C0 0D 62 C0 10 0C AD 53 91
1710: 81 C9 FF D0 05 A9 01 8D DC
1718: 85 81 A2 01 20 1E FB C0 FF
1720: C0 90 06 A9 01 8D 6B 1D 60
1728: 60 C0 40 90 06 A9 00 8D 2C
1730: 6B 1D 60 A9 FF 8D 6B 1D 2C
1738: 60 20 FE 15 AD 43 80 C9 15
1740: 20 F0 13 20 E0 1A 20 94 63
1748: 12 A9 20 8D 43 80 A9 40 76
1750: 8D 42 80 AD 54 C0 20 58 FE
1758: FC AD 53 C0 A9 15 20 5B 24
1760: FB A2 4C A0 88 20 48 18 36
1768: AD 10 C0 AD 00 C0 30 0B D2
1770: AD 61 C0 0D 62 C0 10 F3 E0
1778: 4C 19 10 8D 10 C0 C9 D1 D6
1780: F0 07 C9 C3 F0 09 4C 19 BB
1788: 10 AD 51 C0 4C 00 E0 20 A4
1790: 58 FC AD 51 C0 A0 BB A2 97
1798: 28 20 48 18 A9 0F 85 24 26
17A0: A2 0B 20 48 18 A9 0D 85 72
17A8: 24 A9 03 20 5B FB A0 3C FD
17B0: A2 0F 20 48 18 20 53 18 7C
17B8: C9 B1 F0 04 C9 B2 D0 F5 47
17C0: 20 F0 FD 38 E9 B1 8D 97 47
17C8: 81 A9 08 85 24 A9 05 20 6D
17D0: 5B FB A0 2D A2 18 20 48 90
17D8: 18 20 53 18 C9 D4 F0 26 B0
17E0: C9 CE F0 1F C9 C1 D0 F1 A0
17E8: A0 06 8C 30 81 A0 07 8C A6
17F0: 32 81 A0 24 8C 6D 1D A0 E3
17F8: 1C 8C 6E 1D A0 00 8C 98 AE
1800: 81 F0 1C A0 00 2C A0 01 AD
```

```
1808: 8C 98 81 A0 05 8C 30 81 1B
1810: A0 06 8C 32 81 A0 18 8C 12
1818: 6D 1D A0 17 8C 6E 1D 20 44
1820: F0 FD A9 0A 85 24 A9 07 35
1828: 20 5B FB A0 15 A2 15 20 46
1830: 48 18 20 53 18 C9 CA F0 32
1838: 07 C9 CB D0 F5 A9 01 2C 69
1840: A9 00 8D 88 81 4C 19 10 FE
1848: B9 5B 18 20 F0 FD 88 CA 8C
1850: D0 F6 60 AD 10 C0 AD 00 6C
1858: C0 10 FB 60 BA CB C3 C9 C8
1860: D4 D3 D9 CF CA A0 D2 CF 76
1868: A0 C4 D2 C1 CF C2 D9 C5 93
1870: CB BA C5 C3 C1 A0 D2 CF 30
1878: A0 AC C5 C3 C9 D6 CF CE 31
1880: A0 AC D2 C5 CE C9 C1 D2 D6
1888: D4 BA D3 D2 C5 D9 C1 CC 5F
1890: D0 A0 B2 A0 D2 CF A0 B1 7A
1898: CE C9 C1 C7 C1 A0 D9 C1 5D
18A0: CC D0 A0 CF D4 A0 D9 C5 1F
18A8: CB A0 D2 C5 C8 D4 CF A0 77
18B0: D9 CE C1 A0 D2 CF A0 A0 7B
18B8: A0 A0 A0 A0 83 D4 C9 D5 58
18C0: D1 A0 CF D4 A0 D1 A0 AC 83
18C8: D3 C7 CE C9 D4 D4 C5 D3 A4
18D0: A0 C5 C7 CE C1 C8 C3 A0 02
18D8: CF D4 A0 C3 A0 D3 D3 C5 38
18E0: D2 D0 A0 A0 D2 C5 CB C5 D7
18E8: C5 D3 A0 D4 C1 C5 C8 A0 A9
18F0: D3 D4 CE C5 D3 C5 D2 D0 A2
18F8: A0 D3 CE CF C9 D4 C1 C3 2E
1900: C9 CC D0 D0 C1 A0 C5 CC 5A
1908: D0 D0 C1 A0 D3 A7 A1 C5 5F
1910: D4 D5 D0 CD CF C3 A0 B5 9D
1918: B3 B6 B4 B0 B2 B1 B2 B1 E6
1920: B2 B1 B2 AF B6 B5 B6 B5 19
1928: B6 B0 B2 B1 B2 B1 B2 AF C4
1930: B6 B5 B6 B0 B2 B1 B2 B1 80
1938: B2 B1 B2 AF AE A7 A5 A4 85
1940: A3 22 A1 A0 AD B4 B3 B4 67
1948: B3 B4 AE AC AB AA A9 A8 26
1950: AD B4 B3 B4 AE A6 A5 A4 48
1958: A3 22 A1 A0 AD 00 20 94 65
1960: 18 A2 02 20 94 18 20 87 56
1968: 15 20 EB 13 E6 16 D0 02 0F
1970: E6 17 A5 10 C5 16 A5 11 74
1978: E5 17 90 0C A5 12 C5 0E 45
1980: 4C 8D 19 43 45 34 00 20 AE
1988: 20 34 32 7F 20 A9 70 8D 2C
1990: 8B 19 A9 00 8D 89 19 AD 76
```

```
1998: 85 19 CD 8B 19 90 04 ED 47
19A0: 8B 19 38 2E 89 19 4E 8B A1
19A8: 19 90 EF C9 04 90 02 69 F5
19B0: FB 2A 8D 8A 19 4A 2E 89 9D
19B8: 19 AD 87 19 0A AD 86 19 92
19C0: 2A 8D 88 19 85 A1 AD 83 9F
19C8: 19 85 F8 AD 84 19 85 F9 70
19D0: AD 8A 19 0A A8 B1 F8 85 C3
19D8: 9F 18 69 02 85 FC C8 B1 91
19E0: F8 85 A0 85 FD 90 02 E6 7A
19E8: FD 20 B8 1A 2C 8C 19 30 D0
19F0: 05 CE 8C 19 50 0E 20 52 C9
19F8: 1A A4 9E B1 FE 11 FC 91 18
1A00: FE 88 10 F7 18 A0 01 B1 4E
1A08: 9F 65 FC 85 FC 90 02 E6 72
1A10: FD E6 A1 C6 9D D0 D5 A9 23
1A18: 80 85 FF A9 00 2C 42 80 3E
1A20: 50 02 A9 20 85 FE 2C 42 F6
1A28: 80 50 08 AD 41 80 EE 41 B7
1A30: 80 10 06 AD 40 80 EE 40 66
1A38: 80 0A 0A A8 A5 9F 91 FE C8
1A40: C8 A5 A0 91 FE C8 AD 88 6E
1A48: 19 91 FE C8 AD 89 19 91 31
1A50: FE 60 A5 A1 29 3F A8 B9 3C
1A58: 78 1A 0D 42 80 85 FF A9 D8
1A60: 08 25 A1 F0 02 A9 80 18 F4
1A68: 24 A1 10 02 69 50 50 02 68
1A70: 69 28 6D 89 19 85 FE 60 E6
1A78: 00 04 08 0C 10 14 18 1C 8C
1A80: 00 04 08 0C 10 14 18 1C 94
1A88: 01 05 09 0D 11 15 19 1D 9C
1A90: 01 05 09 0D 11 15 19 1D A4
1A98: 02 06 0A 0E 12 16 1A 1E AC
1AA0: 02 06 0A 0E 12 16 1A 1E B4
1AA8: 03 07 0B 0F 13 17 1B 1F BC
1AB0: 03 07 0B 0F 13 17 1B 1F C4
1AB8: 38 A9 0F ED 88 19 B0 02 40
1AC0: A9 FF 8D 8C 19 A0 00 B1 41
1AC8: 9F 85 9D C8 B1 9F 38 E9 D4
1AD0: 01 85 9E A9 27 ED 89 19 72
1AD8: C5 9E B0 02 85 9E 60 8D C2
1AE0: 4C E5 1A 20 20 A9 00 8D 2F
1AE8: E4 1A A9 80 85 F9 A9 00 BA
1AF0: 2C 42 80 50 02 A9 20 85 5D
1AF8: F8 2C 42 80 50 05 AD 41 38
1B00: 80 10 03 AD 40 80 8D E3 B8
1B08: 1A AD E3 1A D0 03 4C 70 70
1B10: 1B AC E4 1A B1 F8 85 9F 59
1B18: 18 69 02 85 FC C8 B1 F8 B4
1B20: 85 A0 85 FD 90 02 E6 FD 2A
```

```
1B28: C8 B1 F8 8D 88 19 85 A1 7C
1B30: C8 B1 F8 8D 89 19 C8 8C FD
1B38: E4 1A 20 B8 1A 2C 8C 19 AA
1B40: 30 05 CE 8C 19 50 10 20 BC
1B48: 52 1A A4 9E B1 FC 49 FF C0
1B50: 31 FE 91 FE 88 10 F5 18 89
1B58: A0 01 B1 9F 65 FC 85 FC 76
1B60: 90 02 E6 FD E6 A1 C6 9D 05
1B68: D0 D3 CE E3 1A 4C 09 1B 43
1B70: A9 00 2C 42 80 50 04 8D FF
1B78: 41 80 60 8D 40 80 60 52 6B
1B80: 4C 8A 1B 4C F2 1B 3A 20 40
1B88: 20 3A A2 00 BD 30 81 8D F0
1B90: 86 1B BD 18 81 20 83 1B B8
1B98: 18 BD 01 81 6D 87 1B 9D DF
1BA0: 01 81 BD 00 81 6D 88 1B 5D
1BA8: C9 8C 90 0B 2C 88 1B 30 93
1BB0: 04 E9 8C B0 02 69 8B 9D 6A
1BB8: 00 81 BD 30 81 8D 86 1B 74
1BC0: BD 24 81 20 83 1B 18 BD 87
1BC8: 0D 81 6D 87 1B 9D 0D 81 F6
1BD0: BD 0C 81 6D 88 1B C9 60 94
1BD8: 90 0B 2C 88 1B 10 04 A9 F2
1BE0: 00 F0 02 A9 5F 9D 0C 81 39
1BE8: E8 E8 E0 0C F0 03 4C 8C 63
1BF0: 1B 60 8D 87 1B 8D 89 1B 34
1BF8: 30 03 A9 00 2C A9 FF 8D D2
1C00: 88 1B A0 08 88 0E 86 1B 7C
1C08: 90 FA C0 00 F0 25 0E 87 1F
1C10: 1B 2E 88 1B 0E 86 1B 90 75
1C18: 17 18 AD 87 1B 6D 89 1B CC
1C20: 8D 87 1B 90 03 EE 88 1B 6D
1C28: AD 89 1B 10 03 CE 88 1B 7D
1C30: 88 D0 DB 60 49 4E 45 23 93
1C38: 20 00 20 3C 4C 49 4E 45 B1
1C40: 4C 4A 1C 20 20 20 03 20 5E
1C48: 20 20 AD 00 81 38 ED 02 19
1C50: 81 90 02 18 24 38 6A 8D 93
1C58: 44 1C AD 0C 81 38 ED 0E 07
1C60: 81 90 02 18 24 38 6A 8D A3
1C68: 45 1C AD 44 1C 30 08 C9 BF
1C70: 24 90 0D E9 46 90 06 C9 69
1C78: DE B0 05 69 46 8D 44 1C 90
1C80: A9 00 2C 26 81 30 09 2C B0
1C88: 1A 81 10 0E A9 01 D0 0A 0E
1C90: 2C 1A 81 10 03 A9 02 2C 85
1C98: A9 03 8D 48 1C A9 00 2C 50
1CA0: 45 1C 30 09 2C 44 1C 10 D3
1CA8: 0E A9 01 D0 0A 2C 44 1C 25
1CB0: 10 03 A9 02 2C A9 03 8D A2
```

```
1CB8: 49 1C AD 49 1C 38 ED 48 CC
1CC0: 1C 4A 90 0C 29 01 D0 02 5D
1CC8: A9 FF 8D 43 1C 4C 66 1D B7
1CD0: F0 12 A9 00 38 ED 44 1C 59
1CD8: 8D 44 1C A9 00 38 ED 45 09
1CE0: 1C 8D 45 1C A9 00 8D 46 A3
1CE8: 1C 8D 47 1C AD 44 1C 0D 01
1CF0: 45 1C D0 06 8D 43 1C 4C 4B
1CF8: 66 1D AD 46 1C 18 6D 44 26
1D00: 1C 8D 46 1C AD 47 1C 18 11
1D08: 6D 45 1C 8D 47 1C AD 46 F2
1D10: 1C 38 ED 1A 81 4D 1A 81 BC
1D18: 10 0E AD 47 1C 38 ED 26 CB
1D20: 81 4D 26 81 30 D4 10 1B 5B
1D28: AD 47 1C 38 ED 26 81 4D 6A
1D30: 26 81 30 0C AC 48 1C CC 30
1D38: 49 1C D0 04 A9 00 F0 23 CA
1D40: A9 01 2C A9 00 2C 44 1C 05
1D48: 10 02 49 01 2C 45 1C 10 03
1D50: 02 49 01 AC 48 1C CC 49 5E
1D58: 1C F0 02 49 01 09 00 D0 AE
1D60: 02 A9 FF 8D 43 1C 60 00 2A
1D68: 4C 72 1D 20 44 33 20 20 5A
1D70: 30 34 AD 6B 1D F0 1F 8D B4
1D78: 6F 1D AD 6D 1D 8D 71 1D 5D
1D80: A2 00 2C 6F 1D 30 09 20 64
1D88: BB 1D 20 F6 1D 4C 96 1D BF
1D90: 20 F6 1D 20 BB 1D AD 6C 58
1D98: 1D F0 1F 8D 6F 1D AD 6E 14
1DA0: 1D 8D 71 1D A2 02 2C 6F B1
1DA8: 1D 30 09 20 BB 1D 20 F6 2A
1DB0: 1D 4C BA 1D 20 F6 1D 20 EC
1DB8: BB 1D 60 AD 71 1D 8D 86 A0
1DC0: 1B BD 18 81 20 83 1B AD 06
1DC8: 6F 1D 30 16 38 BD 25 81 ED
1DD0: ED 87 1B 9D 25 81 BD 24 EF
1DD8: 81 ED 88 1B 9D 24 81 4C DE
1DE0: F5 1D 18 BD 25 81 6D 87 CD
1DE8: 1B 9D 25 81 BD 24 81 6D C3
1DF0: 88 1B 9D 24 81 60 AD 71 86
1DF8: 1D 8D 86 1B BD 24 81 20 49
1E00: 83 1B AD 6F 1D 30 16 18 5F
1E08: BD 19 81 6D 87 1B 9D 19 6D
1E10: 81 BD 18 81 6D 88 1B 9D F8
1E18: 18 81 4C 30 1E 38 BD 19 B3
1E20: 81 ED 87 1B 9D 19 81 BD 4D
1E28: 18 81 ED 88 1B 9D 18 81 18
1E30: 60 80 A0 A0 A0 A0 A0 A0 44
1E38: 4C 47 1E 4C A1 1E 53 20 41
1E40: 20 45 20 20 4E 4E 53 20 56
```

```
1E48: BF 1E AC 3E 1E B1 FC 8D A4
1E50: 00 82 A9 00 8D 01 82 A9 81
1E58: 01 8D 45 1E A9 00 AE 45 F2
1E60: 1E 9D 01 82 BD 3E 1E A8 27
1E68: B1 FC AC 45 1E 0A 10 03 E2
1E70: 38 29 7F 3E 01 82 88 D0 DA
1E78: F4 1D 00 82 9D 00 82 EE 7F
1E80: 45 1E E0 04 D0 D6 A0 05 6B
1E88: B9 00 82 91 F8 91 FA 88 97
1E90: 10 F6 20 DE 1E AC 46 1E D2
1E98: C8 8C 46 1E C0 05 D0 AA 6D
1EA0: 60 20 BF 1E AC 3E 1E B1 3B
1EA8: FC 0A A0 00 91 F8 91 FA 88
1EB0: 20 DE 1E AC 46 1E C8 8C 0C
1EB8: 46 1E C0 05 D0 E6 60 A9 94
1EC0: 00 8D 46 1E A9 F5 85 FC 38
1EC8: A9 1E 85 FD AD 43 1E 09 B1
1ED0: 80 85 F8 85 FA A9 24 85 72
1ED8: F9 A9 44 85 FB 60 18 A5 94
1EE0: F9 69 04 85 F9 49 60 85 88
1EE8: FB A5 FC 18 69 0A 85 FC 29
1EF0: 90 02 E6 FD 60 2A 08 0A 78
1EF8: 0A 02 2A 02 2A 2A 2A 22 90
1F00: 08 20 20 22 02 02 20 22 EA
1F08: 22 22 08 08 08 2A 0A 2A 88
1F10: 08 2A 2A 22 08 02 20 20 EC
1F18: 20 22 08 22 20 2A 08 2A F5
1F20: 0A 20 0A 2A 08 2A 20 1F 97
1F28: AA 80 8A A2 A2 8A 82 82 1E
1F30: D5 80 84 84 84 84 84 94 26
1F38: AA 80 88 A2 A2 AA A2 A2 CE
1F40: D5 80 C4 C4 C4 90 90 90 88
1F48: AA 80 A8 88 88 A8 88 A8 3A
1F50: D5 80 94 C4 C4 94 C4 C4 3F
1F58: AA 80 A0 A0 A0 A0 A0 A0 93
1F60: AA 80 88 A0 A0 A8 88 A8 90
1F68: AA 80 80 80 80 80 88 A0 EB
1F70: D5 80 D1 91 91 D1 91 D0 D4
1F78: AA 80 A8 A0 A0 A0 A0 A0 B4
1F80: D5 80 C1 90 90 C0 80 D0 64
1F88: AA 80 82 80 80 80 82 80 20
1F90: D0 90 90 90 90 90 90 95 F3
1F98: 85 84 84 84 84 84 84 D4 A7
1FA0: 00 2A 20 20 20 20 20 20 51
1FA8: 00 28 08 08 08 08 08 0A EC
1FB0: 00 00 00 00 00 00 00 2A 19
1FB8: 00 00 00 00 00 00 00 55 4C
1FC0: 80 80 80 80 80 80 80 D5 54
1FC8: 80 80 80 80 80 80 80 AA 31
1FD0: 00 2A 00 00 00 00 00 00 99
```

```
1FD8: 00 55 00 00 00 00 00 00 6C
1FE0: 80 80 94 9C 94 80 80 AA 2E
1FE8: 4C F2 1F 00 00 00 00 41 2F
1FF0: 31 20 A9 12 8D EE 1F A9 36
1FF8: 00 85 FE A9 60 85 FF A9 D5
2000: FC 85 FC A9 60 85 FD A9 19
2008: E1 85 F8 A9 20 85 F9 A0 FF
2010: 00 A5 FC 91 FE E6 FE A5 A9
2018: FD 91 FE E6 FE B1 F8 91 4C
2020: FC 8D EB 1F C8 A9 00 8D 2C
2028: EF 1F B1 F8 8D EC 1F 10 5C
2030: 08 29 7F 18 69 01 EE EF 4D
2038: 1F 91 FC C8 8C F0 1F 8C 8B
2040: F1 1F AD EC 1F 29 7F AA 0D
2048: AC F0 1F B1 F8 AC F1 1F 97
2050: 91 FC EE F0 1F EE F1 1F 3D
2058: CA D0 ED AD EF 1F F0 0A B2
2060: A9 80 AC F1 1F 91 FC EE 72
2068: F1 1F CE EB 1F D0 D3 18 FD
2070: AD F0 1F 65 F8 85 F8 90 5E
2078: 02 E6 F9 A9 06 8D ED 1F AE
2080: 18 A5 FC 85 FA AD F1 1F BF
2088: 65 FC 85 FC A5 FD 85 FB 67
2090: 90 02 E6 FD A0 00 A5 FC A3
2098: 91 FE E6 FE A5 FD 91 FE 75
20A0: E6 FE B1 FA 91 FC 8D EB 81
20A8: 1F C8 B1 FA 91 FC 8D EC 19
20B0: 1F C8 AE EC 1F 18 B1 FA 0F
20B8: 2A 30 03 18 09 80 91 FC 66
20C0: C8 CA D0 F2 CE EB 1F D0 96
20C8: E9 CE ED 1F D0 B2 CE EE 3F
20D0: 1F F0 0D 18 98 65 FC 85 D9
20D8: FC 90 02 E6 FD 4C 0F 20 C9
20E0: 60 07 83 83 88 80 87 AA BB
20E8: 80 FF FF 87 F5 FF 8F DD 8E
20F0: EA 8F C0 82 80 D0 80 80 93
20F8: 0D 83 80 80 87 80 E0 87 40
2100: 80 F0 83 80 F8 81 D0 FE A5
2108: 80 D0 AE 80 C0 AF 80 C0 23
2110: AB 80 E0 AB 80 F4 AB 80 CE
2118: DE A0 80 DC 80 80 D0 80 F7
2120: 80 10 82 E0 80 F0 81 F0 C0
2128: 81 F0 81 F0 81 F0 81 F4 6E
2130: 83 F4 8B F5 AB F5 AB F0 BF
2138: 81 F0 81 F0 81 F4 82 F4 90
2140: 8A E0 80 0E 03 8E 80 80 C4
2148: 9E 80 80 BE 80 80 FC 80 76
2150: 80 F8 82 80 F0 AB 80 F0 91
2158: AB 81 D0 8F 80 94 9F 80 F9
2160: 94 BE 80 80 BC 81 80 B8 5A
```

```
2168: 80 80 FA 80 80 B0 80 07 41
2170: 83 80 A0 81 80 A8 80 FE 67
2178: AA 97 FE FF 95 FC FF 9F 16
2180: C0 8A 9C 80 82 98 0D 83 75
2188: 80 A8 80 80 E8 81 90 E8 A4
2190: 83 D0 BE 81 D0 9E 80 D0 8B
2198: 8E 80 D0 8F 80 D0 AB 80 74
21A0: F8 AB 80 FC 80 80 BE 80 2E
21A8: 80 9F 80 80 87 80 80 10 7A
21B0: 82 C0 81 D4 8B D0 8B E0 79
21B8: 83 E0 83 E0 83 F5 AB F5 B4
21C0: AB F4 8B F0 8B E0 83 E0 5E
21C8: 83 E0 83 E0 83 E0 83 C0 EA
21D0: 81 0E 82 8C 80 DE 80 9C 8D
21D8: 80 BD 80 FC A8 F8 A9 F0 18
21E0: 8B D5 8F D4 8F C0 9E 80 DA
21E8: BE 80 FC 80 F8 80 F0 07 05
21F0: 83 D0 80 80 C0 82 80 C8 1B
21F8: F6 BE C2 F6 FE C8 F6 BE F5
2200: C0 82 80 D0 80 80 0F 83 0A
2208: 80 C0 83 80 E0 83 80 F0 3C
2210: 83 80 F0 81 80 D4 80 80 45
2218: 9C 80 80 95 80 C0 87 80 CA
2220: F4 87 80 D4 83 80 D1 82 62
2228: 80 C9 82 80 A8 82 80 C8 90
2230: 80 80 C0 80 80 10 82 98 D6
2238: 80 BC 80 BC 80 BC 80 BC 7C
2240: 80 A8 80 BC 80 BC 80 A8 6B
2248: 80 BC 80 FC 80 D5 82 D5 12
2250: 82 81 82 A9 82 A8 80 0F E7
2258: 83 8E 80 80 9E 80 80 BE D0
2260: 80 80 BC 80 80 A8 81 80 CE
2268: E0 81 80 A0 85 80 80 8F 56
2270: 80 80 BF 81 80 AE 81 80 67
2278: AA 84 80 CA 84 80 D2 80 3C
2280: 80 C8 80 80 88 80 07 83 27
2288: 80 80 85 80 A0 81 BE B7 26
2290: 89 BF B7 A1 BE B7 89 80 03
2298: A0 81 80 80 85 0F 83 80 95
22A0: 88 80 80 C8 80 80 D2 80 12
22A8: 80 CA 84 80 AA 84 80 AE 8F
22B0: 81 80 BF 81 80 8F 80 A0 C9
22B8: 85 80 E0 81 80 A8 81 80 3E
22C0: BC 80 80 BE 80 80 9E 80 43
22C8: 80 8E 80 80 10 82 A8 80 65
22D0: A9 82 81 82 D5 82 D5 82 C9
22D8: FC 80 BC 80 A8 80 BC 80 9C
22E0: BC 80 A8 80 BC 80 BC 80 A2
22E8: BC 80 BC 80 98 80 10 83 B5
22F0: C0 80 80 C8 80 80 A8 82 2C
```

```
22F8: 80 C9 82 80 D1 82 80 D4 B6
2300: 83 80 F4 87 80 C0 87 80 D5
2308: 80 95 80 80 9C 80 80 D4 C8
2310: 80 80 F0 81 80 F0 83 80 3C
2318: E0 83 80 C0 83 80 80 80 6B
2320: 14 04 80 90 80 80 88 95 37
2328: 82 80 A8 D5 8A 80 E8 DD 48
2330: AB 80 A8 F7 AE 80 EA DD 2C
2338: AB 80 BA F7 AE 81 EA FD 9A
2340: BB 81 EA FF AE 81 E8 DD 4D
2348: AB 80 A8 FF AF 81 EA DF D2
2350: BB 81 BA FF AF 81 EA FD 83
2358: BB 81 E8 FF AF 81 EA DF 33
2360: AB 80 BA F7 AE 80 EA DD 9E
2368: AB 80 AA D5 8A 80 A8 D5 D4
2370: 82 80 03 81 8A 8E 8A B1 E5
```

Paratrooper

John Goetz
Translation by Tim Victor

"Paratrooper" is a game of high responsibility—you control the destiny of ten parachutists, giving the go signal that ejects them from the plane. Their safe landings depend on your ability to judge weight factors, windage, and the all-important crucial moment when they should leap. For DOS 3.3 and ProDOS on any Apple II computer.

Almost everyone has seen a parachuting exhibition. Perhaps you've wished that you, too, could fall from the sky on the wings of the wind. The plane drones on, cruising at the proper altitude. You peer out the hatch through wispy remnants of clouds as you decide where to land. You can barely see three tiny squares, far below, surrounded by water. These must be the landing pads, your drop zones. An aquatic landing can lead only to disgrace and severe embarrassment, so you know that you must jump at just the right moment.

There are three different-sized landing pads: The smaller pads promise the greatest honor and reward, but allow less room for error. You know that soon these tiny features will grow at an alarming rate. You consult with the pilot and estimate the perfect moment for your jump by carefully considering your altitude, the speed of the wind, and your own body weight.

Too many late-night pizzas coupled with a low wind speed, and you'll drop like a stone. But if you're a featherweight, and the wind's kicking up, you'll find yourself drifting far afield. With all the facts in, you wait for just the right moment. Then you leap out into the cold, crisp wind—with fingers crossed, of course.

If even reading this description makes you nervous, you'll be glad "Paratrooper" is just a computer game. Rarely is such a simple game so much fun to play. The single-key control and adjustable difficulty levels make this an easy-to-learn, yet challenging, game for young children, too.

Let Your Fingers Do the Jumping

Your plane continuously flies across the screen at an altitude which changes randomly for each jump. The paratroopers' weights and the wind speed change for each jump, too. All this information is displayed on the screen. You have ten paratroopers: ten chances for glory or ten chances for dripping disaster. Each time a jumper lands in the water, he or she is lost to you forever. But if you've calculated the windage and weight properly, the paratrooper lives to jump again.

To send a trooper out the aircraft door, press any key. The three landing pads are worth 25, 50, and 75 points, depending on their size. The smallest drop zone is worth the most points, the largest zone the least. Missing a pad and dropping in the water lowers your total score by 10 points.

Two jump levels are available when you start the game—Easy and Hard. The plane flies faster on the Hard level; the landing pads remain the same size, no matter which level you pick.

Once you've exhausted your stick of ten jumpers, you can play again simply by hitting any key. Tired of all this high-altitude high jinks? Just press the N key at the end of a game to retire to the hangar. There's always another day for jumping.

ProDOS Changes

If you're typing in Paratrooper while using the ProDOS operating system, change lines 200, 220, and 1130 in the program listing to what appears below:

```
200 FOR I = 768 TO I + 85: READ A:X = X + A: POKE
    I,A: NEXT : IF X < > 23417 THEN PRINT "ERROR
    IN DATA STATEMENTS.": STOP
220 PRINT CHR$ (4)"PR# A$300"
1130 DATA 216,133,69,134,70,132,71,166,7
```

Paratrooper

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
8F 100 GOTO 150
6F 110 VTAB AL: HTAB AH: PRINT SK$;
08 120 AH = AH - 1: IF AH = 0 THEN AH = 38
33 130 VTAB AL: HTAB AH: PRINT PL$;
19 140 RETURN
```

```
6A 150 SK$ = "    ":WA$ = "#$%&":PL$ = "'()":TR$(Ø) =
       "*":TR$(1) = "+"
9C 160 P1$ = ",":P2$ = "-":S1$ = ".":S2$ = "/"
18 170 KB = 49152
F4 180 X = Ø: FOR I = 141 * 256 + 24 TO I + 103: REA
       D A:X = X + A: POKE I,A: NEXT
31 190 FOR I = 141 * 256 TO I + 7: POKE I,Ø: NEXT
D7 200 FOR I = 768 TO I + 84: READ A:X = X + A: POKE
       I,A: NEXT : IF X < > 23201 THEN PRINT "ERROR
       IN DATA STATEMENTS.": STOP
56 210 POKE 6,Ø: POKE 7,141
B7 220 POKE 54,Ø: POKE 55,3: CALL 1002
4C 230 HOME : HGR
2B 240 FOR I = 17 TO 20: VTAB I: HTAB 1: FOR J = 1 T
       O 39 STEP 4: PRINT WA$;
C5 250 NEXT : NEXT
9Ø 260 FOR I = 16 TO 17: VTAB I
CC 270 INVERSE : HTAB 6: PRINT " ";: HTAB 20: PRINT
       "  ";: HTAB 35: PRINT "  ";: NEXT
86 280 FOR I = 21 TO 23: HTAB 1: VTAB I: FOR J = Ø T
       O 39: PRINT " ";: NEXT : NEXT
F3 290 NORMAL : VTAB 21: HTAB 2: PRINT "  WIND  ";:
       HTAB 12: PRINT " WEIGHT ";: HTAB 22: PRINT "
       SCORE  ";: HTAB 32: PRINT "TROOPERS";
46 300 GOSUB 730
A6 310 AL = RND (1) * 7 + 1:AH = 39:WD = INT (1 + 1Ø
       * RND (1)):WG = INT (75 + 175 * RND (1))
8A 320 PD = WD / 15:PG = WG / 250
E7 330 VTAB 22: HTAB 4: PRINT "    ";: HTAB 13: PRIN
       T "    ";
2C 340 HTAB 23: PRINT "    ";: HTAB 34: PRINT "
       ";
ØD 350 VTAB 22: HTAB 5: PRINT WD;:: HTAB 14: PRINT W
       G;
35 360 HTAB 24: PRINT SC;: HTAB 35: PRINT TR;
4D 370 POKE 49168,Ø
85 380 GOSUB 110: FOR I = 1 TO DF: NEXT : IF PEEK (K
       B) > 128 THEN POKE 49168,Ø: GOTO 400
A7 390 GOTO 380
35 400 PY = AL + 1:PX = AH + 1
B9 410 GOSUB 110: FOR I = PY TO PY + 1: VTAB I: HTAB
        PX: PRINT TR$(I - PY);: NEXT
97 420 FOR I = 1 TO 80: NEXT
21 430 FOR I = PY TO PY + 1: VTAB I: HTAB PX: PRINT
       " ";: NEXT
8C 440 PX = PX + PD: IF PX > 41 THEN PX = PX - 40
C1 450 PY = PY + PG: IF PY > 14 THEN GOSUB 480: IF P
       Y = Ø THEN 310
F3 460 IF PY > 16 THEN GOSUB 620: GOTO 310
1E 470 GOTO 410
```

```
DD 480 IF PX < 6 THEN RETURN
96 490 IF PX < 7 THEN SC = SC + 75: GOTO 550
01 500 IF PX < 20 THEN RETURN
B1 510 IF PX < 22 THEN SC = SC + 50: GOTO 550
10 520 IF PX < 35 THEN RETURN
33 530 IF PX < 38 THEN SC = SC + 25: GOTO 550
1D 540 RETURN
11 550 FOR I = 14 TO 15: VTAB I: HTAB PX: PRINT TR$(
       I - 14);: NEXT
7D 560 VTAB AL: HTAB AH: PRINT SK$;
C8 570 FOR I = 1 TO 200: NEXT : VTAB 14: HTAB PX: PR
       INT " ";: HTAB PX: PRINT P1$;
07 580 FOR I = 1 TO 200: NEXT : HTAB PX: PRINT " ";:
        HTAB PX: PRINT P2$;
2A 590 VTAB 24: HTAB 2: PRINT "CONGRATULATIONS! MISS
       ION ACCOMPLISHED";: FOR I = 1 TO 1200: NEXT :
        HTAB 1: CALL - 868
B8 600 FOR I = 14 TO 15: VTAB I: HTAB PX: PRINT " ";
       : NEXT
A9 610 PY = 0: RETURN
76 620 VTAB AL: HTAB AH: PRINT SK$;
E0 630 FOR I = 15 TO 16: VTAB I: HTAB PX: PRINT " ";
       : NEXT
F4 640 VTAB 16: HTAB PX: PRINT S1$;: FOR I = 1 TO 20
       0: NEXT : VTAB 16: HTAB PX: PRINT " ";
D1 650 VTAB 16: HTAB PX: PRINT S2$;: VTAB 24: HTAB 2
       : PRINT "SPLASH! PARATROOPER MISSED THE TARGE
       T";
CD 660 FOR I = 1 TO 1200: NEXT : HTAB 1: CALL - 868:
        VTAB 16: HTAB PX: PRINT " ";
2C 670 SC = SC - 10: IF SC < 0 THEN SC = 0
FF 680 TR = TR - 1: IF TR > 0 THEN RETURN
04 690 VTAB 22: HTAB 24: PRINT SC;: HTAB 35: PRINT T
       R;
BB 700 VTAB 24: HTAB 2: PRINT "GAME OVER- PRESS ANY
       KEY TO PLAY AGAIN";
BD 710 IF PEEK (KB) < 128 THEN 710
4B 720 POKE 49168,0: VTAB 24: HTAB 1: CALL - 868
08 730 SC = 0:TR = 10
13 740 VTAB 24: HTAB 2: PRINT "SELECT DIFFICULTY: (1
       ) EASY, (2) HARD";
C6 750 IF PEEK (KB) < 128 THEN 750
BE 760 POKE 49168,0: IF PEEK (KB) = 49 THEN DF = 150
       : GOTO 790
76 770 IF PEEK (KB) = 50 THEN DF = 30: GOTO 790
A8 780 GOTO 750
CC 790 VTAB 24: HTAB 1: CALL - 868: RETURN
56 1000 DATA 145,196,145,196,145,196,145,196
37 1010 DATA 162,136,162,136,162,136,162,136
```

```
13 1020 DATA 196,145,196,145,196,145,196,145
E4 1030 DATA 136,162,136,162,136,162,136,162
9F 1040 DATA 0,0,0,252,255,255,0,0
D5 1050 DATA 0,134,143,255,255,255,252,224
8E 1060 DATA 192,224,240,255,255,191,0,0
3A 1070 DATA 190,255,227,227,162,162,162,156
10 1080 DATA 156,136,255,156,156,148,148,148
8A 1090 DATA  0,0,190,255,227,227,162,156
1E 1100 DATA  0,0,0,0,0,0,0,156
EC 1110 DATA 190,255,227,227,156,156,136,255
14 1120 DATA 0,0,0,0,190,255,227,227
0B 1130 DATA 133,69,134,70,132,71,166,7
0B 1140 DATA 10,10,176,4,16,62,48,4
79 1150 DATA 16,1,232,232,10,134,27,24
94 1160 DATA 101,6,133,26,144,2,230,27
BA 1170 DATA 165,40,133,8,165,41,41,3
20 1180 DATA 5,230,133,9,162,8,160,0
EC 1190 DATA 177,26,36,50,48,2,73,127
31 1200 DATA 164,36,145,8,230,26,208,2
D7 1210 DATA 230,27,165,9,24,105,4,133
07 1220 DATA 9,202,208,226,165,69,166,70
72 1230 DATA 164,71,76,240,253
```

Webster Dines Out

Walter Bulawa
Translation by Tim Victor

Tired of blasting invaders from outer space? This whimsical game is set in a very different world—the miniature jungle in your own backyard. DOS 3.3 or ProDOS.

Guide Webster, the hungry tree spider, in his endless search for a square meal. Roving back and forth across his tree limb, he watches for bugs to appear in the grass below. When the time is right, he drops down on a strand of silk for a light snack, then climbs back up his web to look for more.

Unfortunately, this backyard paradise isn't quite perfect. The more Webster eats, the faster the bugs move, making it harder to find the next meal. Even worse, he's not the only one with an appetite—there's a greedy grasshopper sharing the same hunting ground, stealing bugs when he can and dealing out deathly touches whenever Webster drops too close.

Getting Webster on Disk

"Webster Dines Out" runs on any Apple II-series computer, using either DOS 3.3 or ProDOS. Since it's written entirely in machine language (ML), it must be entered using the "AppleMLX" machine language editor found in Appendix C. AppleMLX greatly simplifies the usually tedious job of accurately entering the mass of numbers that make up an ML program. Be sure you've read Appendix C and have entered and saved AppleMLX to disk before you begin typing in Webster.

When you run AppleMLX, it asks for a starting and ending address. Use these values:

```
START ADDRESS? 1100
END ADDRESS?   1F17
```

AppleMLX then gives you a menu of options. Choose E for enter and type 1100 as the starting address. A prompt for the first line will appear, and you can begin entering the data from the program listing. If you don't type the entire listing in one sitting, follow the instructions in Appendix C for saving a

partially complete version and reloading it later. When you're finished typing, AppleMLX prompts you for a filename for the completed ML program. To load and run Webster Dines Out, simply type BRUN WEBSTER (or whatever name you used for the completed program) and press Return.

Scoring Meals

Use the left- and right-arrow keys to move on the branch, and press the space bar to drop Webster to the ground. Avoid colliding with the giant grasshopper—when that happens, Webster loses a life (and is bodily carried off the screen). The grass in the lawn grows higher and thicker as the game progresses, making your job more difficult. You can drop into the grass to snare a hidden bug, but be sure to keep track of the giant hopper, who might be lurking there as well.

Your goal is to score points as quickly as possible. Each bug is worth 25 points—you get 50 bonus points for snaring two bugs in a single drop. Webster has three lives in each game; getting snatched by the giant hopper costs you a life but doesn't reduce your score.

There are six skill levels, each harder than the last. As you advance to higher levels, the bugs and hopper speed up, and the grass grows longer. The game ends when you lose all three lives or exhaust your time at the highest skill level.

Webster Dines Out

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program.

```
START ADDRESS: 1100
END ADDRESS:   1F17

1100: 20 AC 19 A9 00 8D 0D 08 73
1108: 8D 0E 08 A9 20 8D 1E 08 8B
1110: 85 E6 20 F2 F3 2C 57 C0 A1
1118: 2C 52 C0 2C 50 C0 2C 10 AD
1120: C0 A9 40 85 E6 20 F2 F3 FE
1128: A9 60 85 E6 20 F2 F3 A9 B4
1130: 50 85 FC A9 06 85 FD A9 02
1138: 20 A0 27 91 FC 88 10 FB B6
1140: E6 FD A5 FD C9 08 D0 04 FD
1148: A9 0A 85 FD C9 0C D0 E7 5A
1150: A5 FC 69 7F 90 DB 2C 53 49
1158: C0 A0 04 A9 01 20 78 19 B0
1160: A0 70 A9 1A A2 0B 20 98 DF
```

```
1168: 19 A0 04 A9 02 20 78 19 F4
1170: A0 7B A9 1A A2 0B 20 98 B2
1178: 19 A0 19 A9 01 20 78 19 9F
1180: A0 86 A9 1A A2 0B 20 98 85
1188: 19 A0 19 A9 02 20 78 19 B7
1190: A0 91 A9 1A A2 0B 20 98 58
1198: 19 A9 00 A8 99 B8 08 C8 C4
11A0: C0 78 D0 F8 20 A3 16 20 C6
11A8: 04 17 20 5A 16 A9 00 8D 21
11B0: 18 08 8D 19 08 2C 2E 08 79
11B8: 30 06 20 60 12 4C F6 11 3F
11C0: 20 25 13 B0 26 2C 2F 08 F1
11C8: 10 13 20 3B 14 90 08 20 82
11D0: 58 14 90 17 4C 0B 11 20 78
11D8: E9 14 20 A7 15 2C 2E 08 31
11E0: 10 09 20 39 15 20 72 16 09
11E8: 4C F6 11 A9 00 8D 21 09 2D
11F0: 8D 28 09 8D 2E 08 20 DD 8D
11F8: 12 AD 18 08 F0 17 20 9F D6
1200: 16 AD 18 08 18 6D 11 08 BE
1208: 8D 11 08 90 08 20 4F 13 B3
1210: 90 03 4C 0B 11 AD 19 08 F0
1218: F0 03 20 9B 16 AD 00 C0 5B
1220: 10 3B 2C 10 C0 C9 8D D0 BA
1228: 03 20 9B 15 2C 2E 08 30 F4
1230: 2C C9 88 D0 0A AD 27 09 59
1238: F0 23 CE 27 09 10 1E C9 78
1240: 95 D0 0C AD 27 09 C9 23 D3
1248: F0 13 EE 27 09 10 0E C9 68
1250: A0 D0 0A A9 80 8D 2E 08 73
1258: A9 08 8D 30 08 4C A7 11 D9
1260: A2 00 A0 00 8C 20 08 BD 9C
1268: 08 09 D0 29 20 6F 15 29 91
1270: DE D0 5C 20 6F 15 10 0B C0
1278: A9 01 99 32 08 A0 B5 A9 DF
1280: 1A D0 09 A9 FF 99 32 08 74
1288: A0 BA A9 1A 20 7C 15 EE 8E
1290: 18 08 4C CF 12 B9 32 08 2D
1298: 2C 1E 08 F0 03 70 12 2C 94
12A0: 50 0F 18 7D 09 09 9D 09 3C
12A8: 09 C9 FD 30 1D C9 28 10 F6
12B0: 19 38 BD 09 09 ED 2C 09 19
12B8: 90 15 C9 03 B0 11 AD 2B 24
12C0: 09 F0 0C 20 6F 15 29 03 4E
12C8: D0 05 A9 00 9D 08 09 8A 75
12D0: 18 69 05 AA AC 20 08 C8 65
12D8: C0 04 D0 88 60 AD 2B 09 1A
12E0: D0 2B 2C 2E 08 30 3D 20 3C
12E8: 6F 15 29 DC D0 36 A9 00 AF
12F0: 38 ED 31 08 8D 31 08 30 C4
```

```
12F8: Ø6 AØ AB A9 1A DØ Ø4 AØ 15
13ØØ: BØ A9 1A A2 23 2Ø 7C 15 FD
13Ø8: EE 19 Ø8 DØ 17 AD 2C Ø9 CA
1318: 18 6D 31 Ø8 8D 2C Ø9 C9 3D
1318: F9 3Ø Ø4 C9 28 3Ø Ø5 A9 1A
1328: ØØ 8D 2B Ø9 6Ø AD 28 Ø9 B2
1328: DØ Ø8 AD 3Ø Ø8 1Ø Ø3 38 3Ø
1338: BØ 1C C9 78 9Ø Ø8 AØ F8 55
1338: 8C 3Ø Ø8 AØ 8Ø 2C AØ ØØ B1
1348: 8C 2F Ø8 18 AD 28 Ø9 6D 88
1348: 3Ø Ø8 8D 28 Ø9 18 6Ø A9 CF
1358: DØ 8D 11 Ø8 AC 1Ø Ø8 CØ 5B
1358: B6 DØ Ø3 4C B1 14 C8 8C 2F
1368: 1Ø Ø8 2Ø 49 16 A9 28 8D 5E
1368: 17 Ø8 A9 47 8D 17 18 A9 68
1378: 18 8D 18 18 A9 14 85 FC 3Ø
1378: A9 1B 85 FD 2Ø 6F 15 8D 41
1388: 16 Ø8 C9 AØ BØ F6 4A 4A 37
1388: 29 FE 8D Ø7 Ø8 AD 16 Ø8 5Ø
1398: 29 Ø7 C9 Ø7 FØ E6 18 69 73
1398: Ø2 8D 16 Ø8 29 Ø1 FØ Ø6 9B
13AØ: EE Ø7 Ø8 AØ Ø6 2C AØ Ø2 2F
13A8: B1 FC 85 EE C8 B1 FC 85 13
13BØ: EF C8 B1 FC 85 FE C8 B1 72
13B8: FC 85 FF A9 Ø2 8D 12 Ø8 CB
13CØ: A9 ØØ 8D 13 Ø8 A9 78 8D Ø4
13C8: 14 Ø8 A9 7F 8D 15 Ø8 AC A5
13DØ: 16 Ø8 FØ 24 ØE 12 Ø8 1Ø 3D
13D8: Ø8 AD 12 Ø8 69 8Ø 8D 12 AB
13EØ: Ø8 2E 13 Ø8 38 2E 14 Ø8 24
13E8: 1Ø Ø8 AD 14 Ø8 69 8Ø 8D 84
13FØ: 14 Ø8 2E 15 Ø8 88 DØ DC 1B
13F8: AØ ØB AD 13 Ø8 91 EE AD 2B
14ØØ: 15 Ø8 91 FE 88 AD 12 Ø8 FD
1408: 91 EE AD 14 Ø8 91 FE 88 B8
1410: 1Ø E8 A9 87 8D Ø8 Ø8 2Ø E4
1418: DC 17 AD 1Ø Ø8 C9 B4 9Ø 8C
1420: 1Ø A9 14 85 FC A9 1B 85 DF
1428: FD A9 81 8D Ø8 Ø8 2Ø DC 4Ø
1438: 17 CE 17 Ø8 FØ Ø3 4C 74 9B
1438: 13 18 6Ø AD 27 Ø9 2C 31 BD
1448: Ø8 3Ø Ø3 38 E9 Ø7 CD 2C 8F
1448: Ø9 FØ ØC 9Ø Ø3 18 9Ø Ø7 5C
1458: 69 Ø4 CD 2C Ø9 9Ø ØØ 6Ø 95
1458: A9 ØØ 8D 21 Ø9 A9 78 8D 86
1468: 28 Ø9 18 AD 2C Ø9 6D 31 4E
1468: Ø8 8D 2C Ø9 C9 F9 3Ø 2B CF
1478: C9 28 1Ø 27 2C 31 Ø8 3Ø 62
1478: Ø3 A9 Ø5 2C A9 FE 18 6D D6
1488: 2C Ø9 8D 27 Ø9 C9 FC 3Ø BE
```

```
1488: 04 C9 28 30 05 A9 00 8D 89
1490: 26 09 20 90 16 20 04 17 6B
1498: 4C 58 14 A0 C9 A9 1A A2 55
14A0: 1E 20 7C 15 CE 0F 08 20 A3
14A8: 38 16 AD 0F 08 C9 B0 D0 B2
14B0: 36 A0 0D A9 03 20 78 19 FA
14B8: A0 9C A9 1A A2 0F 20 98 59
14C0: 19 2C 10 C0 AD 00 C0 10 8D
14C8: FB 2C 10 C0 C9 CE F0 04 77
14D0: C9 EE D0 03 4C F1 16 A9 E3
14D8: 20 A0 00 99 D0 07 99 D0 79
14E0: 0B C8 C0 28 D0 F5 60 18 92
14E8: 60 A2 00 8E 21 08 BD 08 7F
14F0: 09 F0 3C BD 09 09 38 E9 04
14F8: 04 CD 27 09 10 31 18 69 EA
1500: 06 CD 27 09 30 29 A9 00 8F
1508: 9D 08 09 AD 21 08 F0 0E 18
1510: 18 AD 0B 08 69 32 8D 0B CD
1518: 08 90 03 EE 0C 08 EE 21 39
1520: 08 18 AD 0B 08 69 19 8D 60
1528: 0B 08 90 03 EE 0C 08 8A 5E
1530: 18 69 05 AA C9 14 D0 B6 03
1538: 60 AD 1E 08 29 20 F0 0C F9
1540: AD 28 09 8D 00 1B A0 BF B2
1548: A9 1A D0 0A AD 28 09 8D 36
1550: 0A 1B A0 C4 A9 1A A2 19 BA
1558: 20 7C 15 AD 27 09 18 69 26
1560: 02 8D 22 09 AD 28 09 F0 D4
1568: 02 A9 01 8D 21 09 60 AD 92
1570: 23 08 0A 0A 38 6D 23 08 D5
1578: 8D 23 08 60 8C 8A 15 8D 7F
1580: 8B 15 A0 04 8A 18 69 04 95
1588: AA B9 FF FF 9D 08 09 CA 60
1590: 88 10 F6 E8 AC 8A 15 AD D7
1598: 8B 15 60 2C 10 C0 2C 00 78
15A0: C0 10 FB 2C 10 C0 60 20 D5
15A8: C5 15 AE 0B 08 EC 0D 08 97
15B0: AD 0C 08 ED 0E 08 90 0C 52
15B8: 8E 0D 08 AD 0C 08 8D 0E F2
15C0: 08 20 D8 15 60 AC 0B 08 37
15C8: AD 0C 08 20 F2 15 A0 0A 07
15D0: A9 01 20 78 19 4C E8 15 7C
15D8: AC 0D 08 AD 0E 08 20 F2 3C
15E0: 15 A0 1F A9 01 20 78 19 CE
15E8: A0 29 A9 08 A2 05 20 98 65
15F0: 19 60 8C 27 08 8D 28 08 92
15F8: A2 04 A9 B0 9D 29 08 CA 22
1600: 10 FA E8 AD 27 08 DD 34 34
1608: 16 AD 28 08 FD 30 16 90 9D
1610: 11 8D 28 08 AD 27 08 FD C5
```

```
1618: 34 16 8D 27 08 FE 29 08 9E
1620: D0 E1 E8 E0 04 D0 DC AD 23
1628: 27 08 69 AF 9D 29 08 60 14
1630: 27 03 00 00 10 E8 64 0A A7
1638: A0 0C A9 02 20 78 19 A0 C2
1640: 0F A9 08 A2 01 20 98 19 5C
1648: 60 A0 21 A9 02 20 78 19 26
1650: A0 10 A9 08 A2 01 20 98 78
1658: 19 60 A9 80 8D 6B 1A A9 5E
1660: 07 8D 0A 08 8D 09 08 A9 7F
1668: 39 8D 6D 1A A9 08 8D 6E DA
1670: 1A 60 AD 28 09 4A 4A 4A 4A
1678: 69 06 8D 0A 08 8D 09 08 BD
1680: A9 80 8D 6B 1A A9 38 8D 7F
1688: 6D 1A A9 08 8D 6E 1A 60 62
1690: AD 1E 08 29 20 F0 C3 A9 A4
1698: 0A D0 DF A9 24 D0 DB A9 5A
16A0: 06 D0 D7 A9 B3 8D 0F 08 93
16A8: 20 38 16 A9 B1 8D 10 08 3C
16B0: A9 D0 8D 11 08 20 49 16 12
16B8: A9 00 8D 0B 08 8D 0C 08 B2
16C0: 20 C5 15 20 D8 15 A0 00 6F
16C8: 98 99 08 09 C8 C0 28 D0 A3
16D0: F8 A9 00 A0 7F 99 38 08 C8
16D8: 88 D0 FA A9 80 8D 38 08 2A
16E0: 20 5A 16 A9 01 8D 31 08 B9
16E8: A0 C9 A9 1A A2 1E 4C 7C 51
16F0: 15 2C 54 C0 2C 51 C0 AD 1F
16F8: 00 BE C9 4C F0 03 4C D0 CF
1700: 03 4C 00 BE AD 1E 08 49 ED
1708: 60 8D 1E 08 29 20 F0 02 BB
1710: A9 01 AA BD 54 C0 A9 38 B5
1718: 8D 17 18 A9 18 8D 18 18 AF
1720: A9 00 8D 1F 08 8D 22 08 89
1728: 0A 0A 6D 22 08 8D 22 08 70
1730: AD 1E 08 29 40 F0 02 A9 C3
1738: 28 18 6D 22 08 AA BD B8 6F
1740: 08 F0 1E BD BB 08 85 FC 54
1748: BD BC 08 85 FD BD B9 08 40
1750: 8D 07 08 BD BA 08 8D 08 FC
1758: 08 20 DC 17 A9 00 9D B8 E0
1760: 08 20 8C 18 D0 BF A9 1F A5
1768: 8D 17 18 A9 18 8D 18 18 FF
1770: A9 00 8D 1F 08 8D 22 08 D9
1778: 0A 0A 6D 22 08 AA BD 08 6C
1780: 09 D0 03 4C D6 17 BD 0B 26
1788: 09 85 FC BD 0C 09 85 FD A5
1790: BD 09 09 8D 07 08 BD 0A B7
1798: 09 8D 08 08 AD 1F 08 8D B7
17A0: 22 08 0A 0A 6D 22 08 8D 55
```

```
17A8: 22 08 AD 1E 08 29 40 F0 D7
17B0: 02 A9 28 18 6D 22 08 AA 7F
17B8: A9 01 9D B8 08 A5 FC 9D A9
17C0: BB 08 A5 FD 9D BC 08 AD 01
17C8: 07 08 9D B9 08 AD 08 08 DA
17D0: 9D BA 08 20 DC 17 20 8C 8F
17D8: 18 D0 9A 60 A0 00 B1 FC 06
17E0: 8D 1B 08 C8 B1 FC 8D 1C E2
17E8: 08 8D 1D 08 AD 08 08 8D CD
17F0: 1A 08 AD 07 08 29 01 0A 45
17F8: 0A 69 02 A8 B1 FC 85 EE CC
1800: C8 B1 FC 85 EF C8 B1 FC FB
1808: 85 FE C8 B1 FC 85 FF 20 0D
1810: CF 18 AC 1D 08 88 20 FF 38
1818: FF 20 6D 18 D0 F1 60 B1 40
1820: EC 11 FE 31 FC 8D 2F 18 92
1828: B1 EC 49 7F 31 EE 09 00 E4
1830: 91 FC 20 95 18 88 10 E7 B0
1838: B1 EC 11 FE 31 FC 91 FC 2C
1840: 20 95 18 88 10 F2 60 A5 24
1848: FC 8D 5C 18 A5 FD 49 60 7F
1850: 8D 5D 18 B1 FC 31 EC 11 54
1858: EE 91 FC 99 FF FF B1 FE FF
1860: 49 7F 11 EC 91 EC 20 95 1C
1868: 18 88 10 E7 60 18 AD 1C 22
1870: 08 65 EE 85 EE 90 02 E6 D8
1878: EF 18 AD 1C 08 65 FE 85 77
1880: FE 90 02 E6 FF EE 1A 08 FA
1888: CE 1B 08 60 EE 1F 08 AD 9F
1890: 1F 08 C9 08 60 20 6A 1A 7E
1898: F0 03 2C 30 C0 4A 90 0A E4
18A0: EE 6D 1A D0 03 EE 6E 1A BE
18A8: A9 80 8D 6B 1A CE 09 08 5C
18B0: D0 1C A9 80 8D 6B 1A AD 89
18B8: 0A 08 8D 09 08 4A 4A 4A 7A
18C0: 38 49 FF 6D 6D 1A 8D 6D 92
18C8: 1A B0 03 CE 6E 1A 60 AD C9
18D0: 07 08 10 27 A5 EE 38 ED 42
18D8: 07 08 85 EE 90 02 E6 EF 78
18E0: A5 FE 38 ED 07 08 85 FE EB
18E8: 90 02 E6 FF AD 1D 08 18 C8
18F0: 6D 07 08 8D 1D 08 A9 00 CF
18F8: 8D 07 08 38 A9 28 ED 07 07
1900: 08 CD 1D 08 B0 03 8D 1D 97
1908: 08 AD 1A 08 29 3F A8 B9 BE
1910: 38 19 0D 1E 08 85 FD 09 83
1918: 60 85 ED AD 1A 08 29 08 BF
1920: C9 08 A9 00 6A 2C 1A 08 AE
1928: 10 02 69 50 50 02 69 28 9A
1930: 6D 07 08 85 FC 85 EC 60 6C
```

```
1938: 00 04 08 0C 10 14 18 1C 4A
1940: 00 04 08 0C 10 14 18 1C 52
1948: 01 05 09 0D 11 15 19 1D 5A
1950: 01 05 09 0D 11 15 19 1D 62
1958: 02 06 0A 0E 12 16 1A 1E 6A
1960: 02 06 0A 0E 12 16 1A 1E 72
1968: 03 07 0B 0F 13 17 1B 1F 7A
1970: 03 07 0B 0F 13 17 1B 1F 82
1978: 8C A3 19 18 6A 6A 08 18 A3
1980: 69 50 6D A3 19 8D A3 19 C2
1988: 8D A6 19 28 A9 03 2A 8D 0C
1990: A4 19 69 04 8D A7 19 60 66
1998: CA 8C A0 19 8D A1 19 BD DB
19A0: FF FF 9D FF FF 9D FF FF FC
19A8: CA 10 F4 60 A9 CE 85 FC 79
19B0: A9 1A 85 FD A9 00 85 FE 26
19B8: A9 0C 85 FF A9 08 8D 26 22
19C0: 08 A0 00 B1 FC 8D 1B 08 96
19C8: C8 B1 FC 8D 1C 08 C8 A5 7C
19D0: FE 91 FC C8 A5 FF 91 FC 60
19D8: A0 06 B1 FC 85 EE C8 B1 0E
19E0: FC 85 EF AC 1C 08 88 B1 7F
19E8: EE 4A 91 FE 88 30 03 20 72
19F0: 5D 1A 20 44 1A CE 1B 08 EA
19F8: D0 E9 A0 00 B1 FC 8D 1B D9
1A00: 08 A0 04 A5 FE 91 FC C8 3C
1A08: A5 FF 91 FC A0 08 B1 FC 96
1A10: 85 EE C8 B1 FC 85 EF AC B1
1A18: 1C 08 88 B1 EE 09 80 4A 6F
1A20: 91 FE 88 30 03 20 5D 1A 5E
1A28: 20 44 1A CE 1B 08 D0 E7 30
1A30: 18 A5 FC 69 0A 85 FC 90 01
1A38: 02 E6 FD CE 26 08 F0 03 0A
1A40: 4C C1 19 60 18 AD 1C 08 EB
1A48: 65 EE 85 EE 90 02 E6 EF D4
1A50: 18 AD 1C 08 65 FE 85 FE 31
1A58: 90 02 E6 FF 60 B1 EE 90 6A
1A60: 02 09 80 4A 91 FE 88 10 36
1A68: F4 60 A9 00 2C FF FF 60 26
1A70: D3 C3 CF D2 C5 BA A0 A0 A1
1A78: A0 A0 A0 CC C9 D6 C5 D3 0F
1A80: BA A0 A0 A0 A0 A0 C8 C9 3B
1A88: C7 C8 A0 BA A0 A0 A0 A0 FB
1A90: A0 CC C5 D6 C5 CC BA A0 E5
1A98: A0 A0 A0 A0 D0 D2 C5 D3 94
1AA0: D3 A0 CE A0 D4 CF A0 D1 C3
1AA8: D5 C9 D4 01 F9 80 CE 1A 6E
1AB0: 01 27 80 D8 1A 01 FD 85 23
1AB8: E2 1A 01 27 85 EC 1A 01 8C
1AC0: 00 04 00 1B 01 00 04 0A C1
```

```
1AC8: 1B 01 12 00 F6 1A 0D 08 4F
1AD0: 00 00 00 00 1E 1B 86 1B 8A
1AD8: 0D 08 00 00 00 00 EE 1B 8E
1AE0: 56 1C 08 04 00 00 00 00 88
1AE8: BE 1C DE 1C 08 04 00 00 71
1AF0: 00 00 FE 1C 1E 1D 13 05 57
1AF8: 00 00 00 00 3E 1D 9D 1D EB
1B00: 80 01 00 00 00 00 FC 1D CD
1B08: 7C 1E 80 01 00 00 00 00 24
1B10: FC 1D 7C 1E 06 02 00 00 B5
1B18: 00 00 FC 1E 08 1F 00 00 8C
1B20: 00 00 00 00 00 08 00 00 76
1B28: 00 00 00 00 00 02 00 00 66
1B30: 00 00 00 00 40 00 60 7F A8
1B38: 7F 7F 7F 7F 13 00 00 7E 0D
1B40: 7F 7F 7F 7F 05 00 00 78 9E
1B48: 7F 7F 7F 7F 1D 00 00 2A 19
1B50: 7F 7F 7F 3F 3D 00 00 28 1C
1B58: 75 7F 7F 3F 7D 00 00 00 F8
1B60: 55 7E 7F 2F 75 02 00 00 77
1B68: 40 2A 55 2A 55 02 00 00 49
1B70: 40 02 05 0A 54 00 00 00 2B
1B78: 50 00 05 0A 00 00 00 00 18
1B80: 14 20 01 28 00 00 7F 7F E9
1B88: 7F 7F 7F 7F 7F 63 7F 7F 4E
1B90: 7F 7F 7F 7F 7F 78 7F 7F AA
1B98: 7F 7F 7F 7F 1F 7E 0F 00 67
1BA0: 00 00 00 00 40 7F 7F 00 D5
1BA8: 00 00 00 00 70 7F 7F 03 62
1BB0: 00 00 00 00 40 7F 7F 00 E5
1BB8: 00 00 00 00 00 7F 7F 03 EE
1BC0: 00 00 00 00 00 7E 7F 3F 2F
1BC8: 00 00 00 00 00 78 7F 7F 5F
1BD0: 1F 00 00 00 00 78 7F 7F F6
1BD8: 1F 38 70 60 01 7E 7F 7F 41
1BE0: 07 3E 70 60 7E 7E 7F 7F AA
1BE8: 41 0F 7C 03 7F 7F 40 00 BD
1BF0: 00 00 00 00 00 00 00 02 29
1BF8: 00 00 00 00 00 00 00 08 37
1C00: 00 00 00 00 00 00 00 20 58
1C08: 7E 7F 7F 7F 7F 1F 00 00 BF
1C10: 7D 7F 7F 7F 7F 03 00 60 37
1C18: 7D 7F 7F 7F 7F 00 00 70 43
1C20: 75 7F 7F 7F 57 02 00 78 16
1C28: 75 7F 7F 3F 55 00 00 3A C3
1C30: 55 7F 7F 2B 05 00 00 2A E7
1C38: 55 2A 55 0A 00 00 00 28 19
1C40: 41 02 05 0A 00 00 00 00 DA
1C48: 40 02 05 28 00 00 00 00 44
1C50: 50 00 14 20 01 00 1F 7E F9
```

```
1C58: 7F 7F 7F 7F 7F 7F 7F 78 89
1C60: 7F 7F 7F 7F 7F 7F 7F 63 7C
1C68: 7F 7F 7F 7F 7F 7F 7F 0F 30
1C70: 00 00 00 00 00 40 7F 3F E7
1C78: 00 00 00 00 00 78 7F 0F A0
1C80: 00 00 00 00 00 7E 7F 07 B8
1C88: 00 00 00 00 00 78 7F 03 A4
1C90: 00 00 00 00 00 7E 7F 00 C1
1C98: 00 00 00 00 70 7F 7F 00 51
1CA0: 00 00 00 60 7F 7F 7F 03 DA
1CA8: 18 38 70 60 7F 7F 7F 7F 87
1CB0: 1F 38 70 03 7F 7F 7F 7F 3D
1CB8: 07 7E 41 0F 7C 7F 00 00 0F
1CC0: 00 20 00 00 00 08 58 3B 0D
1CC8: 77 06 40 3B 77 0E 00 3A 28
1CD0: 77 0E 00 20 44 20 00 08 F4
1CD8: 44 00 06 02 01 02 7F 7F E1
1CE0: 7F 0F 7F 7F 7F 63 03 00 14
1CE8: 00 70 0F 00 00 60 7F 00 9F
1CF0: 00 60 7F 0F 11 0E 7F 63 45
1CF8: 11 7F 7F 38 7C 78 04 00 DA
1D00: 00 00 10 00 00 00 60 6E 6B
1D08: 5D 1B 70 6E 5D 03 70 6E F2
1D10: 5D 00 04 22 04 00 00 22 DD
1D18: 10 00 40 00 41 00 71 7F CE
1D20: 7F 7F 47 7F 7F 7F 0F 00 F2
1D28: 00 40 07 00 00 78 07 00 43
1D30: 00 7E 71 08 71 7F 7F 08 49
1D38: 47 7F 1F 3F 1C 7E 00 40 E8
1D40: 03 1C 00 00 60 06 36 00 8A
1D48: 00 30 0C 63 00 00 1E 38 BA
1D50: 43 07 00 3B 6E 65 0D 40 05
1D58: 6D 3A 37 1B 40 4D 6E 15 99
1D60: 1B 60 0C 3A 07 33 60 1E 49
1D68: 6E 45 37 00 37 3A 67 0E 91
1D70: 40 61 6E 35 18 40 41 3A C2
1D78: 17 18 60 00 6E 05 30 60 98
1D80: 60 39 3B 30 00 38 20 60 25
1D88: 01 00 0C 28 01 03 00 06 61
1D90: 28 01 06 00 03 20 00 0C 84
1D98: 40 01 08 01 18 7F 1F 78 B9
1DA0: 41 7F 7F 0F 70 00 7F 7F 3E
1DA8: 07 21 08 7E 7F 40 03 18 B2
1DB0: 70 3F 00 00 00 60 1F 00 B2
1DB8: 00 00 40 1F 00 00 00 40 2D
1DC0: 0F 60 00 30 00 0F 40 00 5A
1DC8: 10 00 3F 00 00 00 60 1F D2
1DD0: 0C 00 00 43 1F 1C 00 40 EE
1DD8: 43 0F 7E 00 70 07 0F 0E 14
1DE0: 00 00 07 7F 03 0F 0F 7C E2
```

```
1DE8: 7F 61 03 3C 78 7F 70 03 05
1DF0: 7C 70 3F 78 0F 7F 61 1F 4D
1DF8: 7C 03 7C 43 20 20 20 20 D7
1E00: 20 20 20 20 20 20 20 20 3C
1E08: 20 20 20 20 20 20 20 20 44
1E10: 20 20 20 20 20 20 20 20 4C
1E18: 20 20 20 20 20 20 20 20 54
1E20: 20 20 20 20 20 20 20 20 5C
1E28: 20 20 20 20 20 20 20 20 64
1E30: 20 20 20 20 20 20 20 20 6C
1E38: 20 20 20 20 20 20 20 20 74
1E40: 20 20 20 20 20 20 20 20 7C
1E48: 20 20 20 20 20 20 20 20 84
1E50: 20 20 20 20 20 20 20 20 8C
1E58: 20 20 20 20 20 20 20 20 94
1E60: 20 20 20 20 20 20 20 20 9C
1E68: 20 20 20 20 20 20 20 20 A4
1E70: 20 20 20 20 20 20 20 20 AC
1E78: 20 20 20 20 0F 0F 0F 0F B4
1E80: 0F 0F 0F 0F 0F 0F 0F 0F BC
1E88: 0F 0F 0F 0F 0F 0F 0F 0F C4
1E90: 0F 0F 0F 0F 0F 0F 0F 0F CC
1E98: 0F 0F 0F 0F 0F 0F 0F 0F D4
1EA0: 0F 0F 0F 0F 0F 0F 0F 0F DC
1EA8: 0F 0F 0F 0F 0F 0F 0F 0F E4
1EB0: 0F 0F 0F 0F 0F 0F 0F 0F EC
1EB8: 0F 0F 0F 0F 0F 0F 0F 0F F4
1EC0: 0F 0F 0F 0F 0F 0F 0F 0F FC
1EC8: 0F 0F 0F 0F 0F 0F 0F 0F 05
1ED0: 0F 0F 0F 0F 0F 0F 0F 0F 0D
1ED8: 0F 0F 0F 0F 0F 0F 0F 0F 15
1EE0: 0F 0F 0F 0F 0F 0F 0F 0F 1D
1EE8: 0F 0F 0F 0F 0F 0F 0F 0F 25
1EF0: 0F 0F 0F 0F 0F 0F 0F 0F 2D
1EF8: 0F 0F 0F 0F 10 00 10 00 F3
1F00: 10 00 10 00 10 00 10 00 E8
1F08: 47 7F 47 7F 47 7F 47 7F F0
1F10: 47 7F 47 7F 00 00 00 00 B2
```

Fill-In-the-Blank Adventure Games

Dale Kroke

Creating your own simple adventure games is a snap when you use the framework provided here. All you have to do is develop your scenario and type in some DATA statements.

Adventure games, those intricate puzzles of logic and mazes of winding tunnels, are among the most popular computer games around. They let players wander through new worlds, where every step may be your last.

But writing your own adventure game may seem complicated, especially if you're just beginning to program on your Apple. I felt the same way myself until I took a closer look at what was required. Then I realized that an adventure game program doesn't have to be complicated or long. That's how Program 1, "Fill-In-the-Blank," came to be written. It provides several standard features of an adventure game, yet it is easy to understand and use. Fill-In-the-Blank is a "generic" program in that it uses no commands which require special translation between computers (such as POKEs or CALLs). Even though it was written on an Apple computer, only minor modifications are needed to run it on other systems.

Features

Fill-In-the-Blank combines complete game control with ease of use. The computer prints your current situation on the screen. To move, you type in the direction (North, South, East, West, Up, or Down), and the computer automatically updates the situation. Other commands include GET *(object)*, which retrieves the object named; USE, which performs some action in the scene; and DROP *(object)*, which places an object into the scene. Typing I (for inventory) temporarily displays the objects you're carrying.

How It Works

Program 1 is actually the main loop for reading and interpreting subsequent DATA statements. The computer reads

through "sets" of data, each representing one situation. The number of sets read is determined by variable L. Here are other variables modified by data:

S1$, S2$—Situation description
T$—Treasure found in the situation
N, S, E, W, U, D—Direction values
V—Value added to L if you USE something
Y—Determines if an enemy is present (value for DROP)

The computer starts with L at a value of 1. It reads and prints S1$ and S2$, giving the first situation. The player then INPUTs a command. If the player wishes to move, the direction value is added to L, a RESTORE is performed, and the computer reads through the DATA until it finds the new situation. (Note: If movement is not permitted in that direction, the variable is assigned a value of 0, and the computer flashes an appropriate message.)

If the player GETs something, then T$ (the treasure) is added to I$ (the inventory). The treasure will be in the room each time the player enters, but it can be taken only once. (Note: If T$="DEATH" it signals the computer that the player has been killed and the program ends.)

The USE and DROP commands are controlled by variables V and Y, which are added to L to send the computer to the modified situation.

How to Use Fill-In-the-Blank

It's called a fill-in-the-blank adventure loop because it's up to you to write the scenario and data to finish the program. Each data set consists of several elements, which must be in this order:

S1$, S2$, N, S, E, W, U, D, V, Y

It doesn't matter how many lines the data is spread over as long as each variable is accounted for.

Program 2 shows the data as it might appear for a playing field in the shape of a 3 × 3 grid. You can also see the method I generally use to arrange the data: S1$ is on one line, with S2$ on the next (sometimes they can be combined on the same line with a comma separating them, as long as they don't exceed the 255-character line-length limit). T$ and the numeric variables appear on the third line of data. Since the computer must read something for every variable, T$ should

be placed equal to X if there's no treasure in a situation (and remember that T$ is equal to DEATH if the character is killed in a situation). The numeric variables should be placed equal to zero if nothing is permitted with that variable. Also remember that you cannot use commas, double-quotes, or colons in the string data. This sometimes makes for odd-looking sentences, but it cannot be avoided.

Values for the numeric variables are determined by how many sets of data must be read through, *from that data set,* before the new situation is described. (That's why you'll see negative numbers in the DATA for Program 2.) When I write long sets of data, I find it easiest to write my opening situation, then write one set of data at a time for each possibility arising from that situation. Then I take the second set and do the same (naturally, there will be fewer possibilities since players can go back the way they came), and so on. You can see that it's possible to get many different branches going off in different directions, but eventually they either meet or tie themselves off.

To see this demonstration game in action, combine Programs 1 and 2 and save them to disk under a new filename, perhaps ADVENTURE. Type RUN ADVENTURE, and you're ready to try out the game.

Using Fill-In-the-Blank with Other Computers

This program was written for an Apple computer, but adapting it for other computers is easy. The HOME command in lines 115, 180, 200, 210, 220, and 235 simply clears the screen and homes the cursor. Owners of different computers can substitute the appropriate command. The GET command in line 225 is similar to INKEY$ on other computers and can be replaced with an INPUT statement if necessary.

By studying the loop in Program 1 and the mini-adventure in Program 2, you should have no problem in writing your own games. Naturally, you can try some improvements such as adding more data to allow sound or hi-res graphics, or creating disk files to allow the computer to retrieve new data and overcome the major limitation on the complexity of the program—the size of the computer's memory. The possibilities go on and on.

Program 1. Fill-In-the-Blank

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
F6 100 REM  *FILL-IN-THE-BLANK ADVENTURE*
62 105 L = 1
B9 110 FOR I = 1 TO L: READ S1$,S2$,T$,N,S,E,W,U,D,V
       ,Y: NEXT I: IF T$ = "DEATH" THEN 235
2A 115 HOME : PRINT S1$;S2$: PRINT : PRINT "WHAT DO
       YOU DO NOW?"
8A 120 INPUT A$:X$ = LEFT$ (A$,1)
7F 125 IF X$ = "N" THEN L = L + N: IF N = 0 THEN 180
22 130 IF X$ = "S" THEN L = L + S: IF S = 0 THEN 180
A2 135 IF X$ = "E" THEN L = L + E: IF E = 0 THEN 180
4E 140 IF X$ = "W" THEN L = L + W: IF W = 0 THEN 180
4D 145 IF X$ = "U" THEN L = L + U: IF U = 0 THEN 180
08 150 IF X$ = "D" THEN L = L + D: IF D = 0 THEN 180
64 155 IF LEFT$ (A$,3) = "GET" THEN 185
7D 160 IF LEFT$ (A$,3) = "USE" THEN 200
FC 165 IF LEFT$ (A$,4) = "DROP" THEN 210
8F 170 IF X$ = "I" THEN 220
81 175 RESTORE : GOTO 110
16 180 HOME : PRINT "YOU CAN'T GO THAT WAY!": FOR Z
       = 1 TO 400: NEXT : GOTO 115
B8 185 IF T$ = "X" THEN PRINT "THAT IS NOT YOURS TO
       TAKE!": GOTO 120
CA 190 X = LEN (T$): FOR Z = 1 TO LEN (I$): IF MID$
       (I$,Z,X) = T$ THEN PRINT "YOU ALREADY HAVE TH
       AT!": GOTO 120
80 192 NEXT Z
07 195 I$ = I$ + "    ":I$ = I$ + T$: GOTO 115
08 200 IF V = 0 THEN HOME : PRINT "YOU CAN'T USE THA
       T HERE!": FOR Z = 1 TO 400: NEXT : GOTO 115
49 205 L = L + V: GOTO 175
B3 210 IF Y = 0 THEN HOME : PRINT "WHY BOTHER?": FOR
        Z = 1 TO 350: NEXT : GOTO 115
CC 215 L = L + Y: GOTO 175
01 220 HOME : PRINT I$: PRINT : PRINT "PRESS ANY KEY
       TO RETURN."
4C 225 GET Z$: IF Z$ = "" THEN 225
9C 230 GOTO 115
AC 235 HOME : PRINT S1$;S2$: PRINT : PRINT "YOU WERE
       KILLED. TRY AGAIN.": END
```

Program 2. Adventure DATA

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
BC 240 REM  *DATA FOR MINI-ADVENTURE*
32 245 REM  *TO BE USED WITH LOOP IN PROGRAM ONE*
6B 250 DATA  YOU ARE IN A FLAT PLAIN LOOKING NORTH.
       THE WAY IS CLEAR IN ALL FOUR DIRECTIONS.
36 255 DATA  YOU CAN BARELY SEE A WALL ENCLOSING THE
        PLAIN ALL AROUND.
66 260 DATA X,1,2,3,4,0,0,0,0
8A 265 DATA  TO THE NORTH IS A WALL. TO THE OTHER DI
       RECTIONS THE WAY IS CLEAR., ON THE GROUND IS
       A SACK OF GOLD COINS.
F1 270 DATA GOLD COINS,0,-1,4,5,0,0,0,0
59 275 DATA  YOU ARE STANDING BY A WALL TO THE SOUTH
       . THE WAY IS CLEAR IN ALL OTHER DIRECTIONS.,
       THERE IS NOTHING ELSE TO SEE.
49 280 DATA X,-2,0,5,6,0,0,0,0
7A 285 DATA  YOU STEP ON A TRAPDOOR AND FALL THROUGH
       ., A POOL OF PIRANHA FISH AWAITS YOU AT THE B
       OTTOM.
D0 290 DATA DEATH,0,0,0,0,0,0,0,0
42 295 DATA  TO THE WEST IS A FEATURELESS WALL. THE
       WAY IS CLEAR IN ALL OTHER DIRECTIONS., ON THE
        GROUND IS A JEWELED SWORD.
3F 300 DATA SWORD,2,4,-4,0,0,0,0,0
76 305 DATA  YOU ARE IN A CORNER WITH WALLS NORTH AN
       D EAST. THERE IS AN INDECIPHERABLE RUNE ON TH
       E EAST WALL. IT LOOKS LIKE IT WAS WRITTEN IN
       BLOOD.
90 310 DATA  TO THE SOUTH AND WEST THE WAY IS OPEN.
DA 315 DATA X,0,-2,0,-4,0,0,0,0
FF 320 DATA  YOU ARE IN A CORNER WITH WALLS TO THE N
       ORTH AND WEST. THERE IS A ROPE LADDER HANGING
        THERE., THE WAY IS CLEAR TO THE SOUTH AND EA
       ST.
AF 325 DATA X,0,-2,-5,0,3,0,0,0
A6 330 DATA  YOU ARE IN THE SOUTH-EAST CORNER. THERE
        IS NOTHING TO SEE., A CLOSER LOOK AT THE WAL
       LS SHOWS A SIGN THAT SAYS ' DON'T GO NORTH '.
D2 335 DATA X,-4,0,0,-5,0,0,0,0
69 340 DATA  YOU ARE IN THE SOUTH-WEST CORNER., THE
       WAY IS CLEAR TO THE NORTH AND EAST.
8D 345 DATA X,-4,0,-5,0,0,0,0,0
E6 350 DATA  THE ROPE BREAKS AS YOU NEAR THE TOP., Y
       OU HIT YOUR HEAD ON THE STONE PAVEMENT AND DI
       E.
DD 355 DATA DEATH,0,0,0,0,0,0,0,0
```

Apple Bowling Champ

Joseph Ganci
Translation by Patrick Parrish

Now you can go bowling without the expense of renting special shoes or suffering the embarrassment of rolling a gutter ball in front of dozens of people. "Apple Bowling Champ" is a game for one to four players which runs on any Apple II–series computer in DOS 3.3 or ProDOS.

Some computer games, such as *Pac-Man* or *Adventure,* create their own unique fantasy worlds, while others are simulations of reality. "Apple Bowling Champ" fits in the second category.

It's not easy to take a game with countless physical variables such as bowling and reduce it to numbers so that it can be re-created by a computer—especially a microcomputer. Compromises must be made. Usually, the game must be modified in major ways to make it possible to program. The result is a hybrid game, an approximation of reality, that resembles the original but has new aspects of its own.

Apple Bowling Champ is a reasonable simulation of a game of tenpins, given the limitations imposed by a BASIC program which must remain short enough to publish in a book. The elements of skill and luck have been preserved, and the scoring is authentic.

Up to Four Players

When you run Apple Bowling Champ, the program asks for the number of players. Up to four people can play. Next, enter the players' names. To fit the names on the 40-column screen, the program truncates entries to eight characters.

Now, you're ready to bowl the first frame. The bowling ball moves rapidly up and down across the alley until you press the space bar. This rolls the ball down the alley and knocks over the pins—unless you've thrown a gutter ball. The trick is to time your release so that the ball rolls down the center of the alley to score a strike.

In case you're unfamiliar with how a game of tenpins is scored, here's a brief summary:

A game consists of ten frames, or turns. Each player gets one or two balls per frame. If you roll a *strike*—knocking down all ten pins with your first ball—you don't get a second ball, but the current ball's score is ten plus the total of your next two throws.

If some pins are left standing after your first ball, you get a second ball. If you knock down all the remaining pins, it counts as a *spare,* and the current ball's score is ten plus your next throw.

If any pins remain after your second ball (no strike or spare), the number of pins knocked down in that frame is added to your previous score.

Rolling a spare in the tenth (last) frame gains you one extra ball; rolling a strike (with your first ball) in the tenth frame gains two extra balls.

Therefore, a perfect game—ten strikes during regular play plus two strikes with the extra balls—scores 300 points. Needless to say, this doesn't happen very often, either in real bowling or in Apple Bowling Champ.

Since Apple Bowling Champ follows every rule of scoring for regular bowling, you can learn how to score by carefully observing the game. The only difference is that the computer doesn't wait until the end of a frame to update the score; it updates it after every ball.

Adjusting the Difficulty

Novice bowlers may find that the ball moves too fast for them to aim. On the other hand, more experienced players may want to speed up the ball to make the game harder. You can easily make either modification by changing the delay loop in line 480. The statement in 480 reads

```
480 FOR R = 1 TO 10: NEXT
```

Replacing the 10 with a larger number slows down the ball; a smaller number speeds up the ball. You might try a value between 20 and 50 for youngsters. For expert players, remove line 480 altogether.

ProDOS

If you're using Apple Bowling Champ with the ProDOS operating system, change lines 130, 830, and 840 in the program listing to what shows below:

```
130 HOME : POKE 230,32: CALL 62450: HGR : POKE 6,
    0: POKE 7,141: PRINT CHR$ (4);"PR# A$300"
830 X = 0: FOR I = 768 TO 853: READ A:X = X + A:
    POKE I,A: NEXT : IF X < > 7950 THEN PRINT "ER
    ROR IN DATA STATEMENTS FOR ML AT 768.": STOP
840 DATA 216,133,69,134,70,132,71,166,7
```

Apple Bowling Champ

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
D5 100 HIMEM: 36096: GOSUB 770
0E 110 GOSUB 830: GOSUB 970
DB 120 GOSUB 1160
7A 130 HOME : POKE 230,32: CALL 62450: HGR : POKE 6,
       0: POKE 7,141: POKE 54,0: POKE 55,3: CALL 100
       2
D5 140 GOSUB 1230
D7 150 REM -MAIN LOOP-
CF 160 FOR Q = 1 TO 10: FOR Z9 = 0 TO A - 1
FA 170 FOR R = 1 TO 10: VTAB 2 * (Z9 + 1) + 1: HTAB
       1: PRINT " ";: FOR F = 1 TO 30: NEXT F: HTAB
       1: PRINT "$": FOR F = 1 TO 30: NEXT F: NEXT R
66 180 B1 = 0: GOSUB 360
63 190 IF J1 < > 10 THEN B1 = 1: GOSUB 390
2C 200 IF Q = 10 THEN ON S GOTO 210,270,270,210,310
CA 210 VTAB (Z9 + 1) * 2 + 1: HTAB 1: PRINT " ": NEX
       T : NEXT : VTAB 24: HTAB 10: POKE - 16368,0:
       PRINT "PLAY AGAIN (Y/N)?";
0F 220 IF PEEK ( - 16384) < 128 THEN 220
02 230 K = PEEK ( - 16384) - 128: IF K < > 78 AND K
       < > 89 THEN POKE - 16368,0: GOTO 220
BB 240 IF K = 89 THEN 120
8A 250 POKE - 16368,0: HOME : TEXT : END
AD 260 REM -10TH FRAME : EXTRA BALLS-
B9 270 VTAB 24: HTAB 5: PRINT "TAKE TWO MORE BALLS,
       "NA$(Z9 + 1);".";
CD 280 FOR I = 1 TO 2000: NEXT : VTAB 24: HTAB 5: PR
       INT SPC( 30);
71 290 S(Z9) = S - 1:B1 = 1: GOSUB 360: IF J < > 10
       THEN 340
98 300 GOTO 330
```

```
2E 310 VTAB 24: HTAB 5: PRINT "TAKE ONE MORE BALL, "
       NA$(Z9 + 1);".";
4B 320 FOR I = 1 TO 2000: NEXT : VTAB 24: HTAB 5: PR
       INT SPC( 29);
FA 330 S(Z9) = 1:B1 = 2: GOSUB 360: GOTO 210
BD 340 S(Z9) = 1:B1 = 2: GOSUB 390: GOTO 210
6E 350 REM -FIRST BALL-
9E 360 FOR I = 1 TO 10: VTAB A(I): HTAB B(I): PRINT
       "*": NEXT
5A 370 PS = 1:J1 = 0: GOTO 400
90 380 REM -SECOND BALL-
91 390 PS = 0
5B 400 GOSUB 450:T = T(Z9):S = S(Z9):T = T + J
33 410 ON S(Z9) GOSUB 660,690,710,730,750
2C 420 T(Z9) = T:S(Z9) = S
59 430 VTAB 21 + (A < 3) + 2 * (Z9 > 1) * (A > 2): H
       TAB 37 - (Z9 / 2 = INT (Z9 / 2)) * 22: PRINT
       T(Z9): RETURN
D4 440 REM -ROLL BALL-
CF 450 H = 1:C = 19:E = 11:D = - 1: POKE - 16368,0
59 460 FOR V = C TO E STEP D: HTAB H: VTAB V: PRINT
       "*";
18 470 IF PEEK ( - 16384) > 127 THEN T5 = V:V = E: N
       EXT : GOTO 510
AD 480 FOR R = 1 TO 10: NEXT
FE 490 HTAB H: PRINT " ";
C5 500 NEXT V:D = - D:T5 = C:C = E:E = T5: GOTO 460
8B 510 V = T5: FOR H = 1 TO 35: HTAB H: VTAB V: PRIN
       T " *";: FOR R = 1 TO 10: NEXT : NEXT
26 520 J = 0
3C 530 IF ( SCRN( H,2 * (V - 1)) + 16 * SCRN( H,2 *
       (V - 1) + 1) - 128) < > 36 THEN 570
3A 540 POKE - 16336,0:J = J + 1: FOR D = - 1 TO 1 ST
       EP 2:X1 = V:X2 = H
C5 550 X1 = X1 + D:X2 = X2 + 1: IF ( SCRN( X2,(X1 -
       1) * 2) + 16 * SCRN( X2,(X1 - 1) * 2 + 1) - 1
       28) = 36 THEN HTAB X2 + 1: VTAB X1: PRINT " "
       ;:J = J + 1: POKE - 16336,0: GOTO 550
09 560 NEXT
47 570 HTAB H: VTAB V: PRINT " *";:H = H + 1: IF H <
        40 THEN 530
DF 580 J1 = J1 + J
E6 590 VTAB 2 * Z9 + 3: HTAB 7 + 3 * Q + B1:G = J +
       48
AE 600 IF J1 < > 10 THEN 630
A4 610 IF PS THEN G = 88: GOTO 630
7D 620 G = 47
9F 630 PRINT CHR$ (G)
26 640 HTAB H: VTAB V: PRINT " ";: RETURN
C5 650 REM -SCORING ROUTINES-
```

```
FD 660 IF J1 < > 10 THEN RETURN
DF 670 IF PS THEN S = 2: RETURN
7E 680 S = 5: RETURN
A5 690 T = T + J: IF J = 10 THEN S = 3: RETURN
4F 700 S = 4: RETURN
54 710 T = T + J * 2: IF J < > 10 THEN S = 4
1B 720 RETURN
C1 730 T = T + J: IF J1 = 10 THEN S = 5: RETURN
F6 740 S = 1: RETURN
7E 750 T = T + J: IF J = 10 THEN S = 2: RETURN
FA 760 S = 1: RETURN
D6 770 DIM A(10),B(10): FOR I = 1 TO 10: READ A(I),B
      (I):X = X + A(I) + B(I): NEXT : IF X < > 540
      THEN PRINT "ERROR IN DATA STATEMENTS FOR PIN
      POSITIONS.": STOP
27 780 RETURN
E7 790 REM -PIN DATA-
E6 800 DATA 12,40,13,39,14,38,14,40
C8 810 DATA 15,37,15,39,16,38,16,40
50 820 DATA 17,39,18,40
46 830 X = 0: FOR I = 768 TO 852: READ A:X = X + A:
      POKE I,A: NEXT : IF X < > 7734 THEN PRINT "ER
      ROR IN DATA STATEMENTS FOR ML AT 768.": STOP
B5 840 DATA 133,69,134,70,132,71,166,7
B5 850 DATA 10,10,176,4,16,62,48,4
EC 860 DATA 16,1,232,232,10,134,27,24
7A 870 DATA 101,6,133,26,144,2,230,27
8D 880 DATA 165,40,133,8,165,41,41,3
40 890 DATA 5,230,133,9,162,8,160,0
93 900 DATA 177,26,36,50,48,2,73,127
C8 910 DATA 164,36,145,8,230,26,208,2
1C 920 DATA 230,27,165,9,24,105,4,133
B3 930 DATA 9,202,208,226,165,69,166,70
69 940 DATA 164,71,76,240,253
23 950 RETURN
50 960 REM LOAD REDEFINED CHARACTERS
AB 970 X = 0:AD = 36096: FOR L = 1 TO 16: READ B: FO
      R I = AD + B TO AD + B + 7: READ A:X = X + A:
       POKE I,A: NEXT :X = X + B: NEXT : IF X < > 6
      223 THEN PRINT "ERROR IN CHARACTER DATA STATE
      MENTS.": STOP
29 980 RETURN
EA 990 DATA 0,0,0,0,0,0,0,0,0
C2 1000 DATA 24,20,20,62,20,62,20,20,0
C7 1010 DATA 32,8,28,8,28,28,62,62,28
44 1020 DATA 80,28,62,127,127,127,62,28,0
7B 1030 DATA 120,63,31,79,103,115,121,124,126
E8 1040 DATA 128,28,34,50,42,38,34,28,0
AF 1050 DATA 136,8,12,8,8,8,8,28,0
```

```
43 1060 DATA 144,28,34,32,24,4,2,62,0
B4 1070 DATA 152,62,32,16,24,32,34,28,0
EA 1080 DATA 160,16,24,20,18,62,16,16,0
71 1090 DATA 168,62,2,30,32,32,34,28,0
6D 1100 DATA 176,56,4,2,30,34,34,28,0
5D 1110 DATA 184,62,32,16,8,4,4,4,0
2A 1120 DATA 192,28,34,34,28,34,34,28,0
5D 1130 DATA 200,28,34,34,60,32,16,14,0
FD 1140 DATA 448,34,34,20,8,20,34,34,0
BA 1150 REM TITLE SCREEN
B4 1160 TEXT : HOME : VTAB 7: HTAB 12: PRINT "BOWLIN
     G CHAMP!": FOR I = 1 TO 4:NA$(I) = "": NEXT
B7 1170 POKE - 16368,0: VTAB 10: HTAB 7: PRINT "HOW
     MANY BOWLERS (1-4): ";
78 1180 IF PEEK ( - 16384) < 128 THEN 1180
94 1190 A = PEEK ( - 16384) - 128: IF A < 49 OR A >
     52 THEN 1160
84 1200 PRINT CHR$ (A):A = A - 48: POKE - 16368,0: F
     OR I = 1 TO A: VTAB 14 + I: HTAB 6: PRINT "B
     OWLER #"I"'S NAME: ";
75 1210 INPUT A$:NA$(I) = LEFT$ (A$,8): NEXT : FOR I
      = 0 TO A - 1:T(I) = 0: NEXT : RETURN
E3 1220 REM DRAW GAME SCREEN
14 1230 VTAB 1: HTAB 10: PRINT "1  2  3  4  5  6  7
     8  9  10"
51 1240 HCOLOR= 3: HPLOT 63,11 TO 279,11
CD 1250 FOR I = 1 TO A: VTAB 2 * I + 1: HTAB 3: PRIN
     T "# "I
FF 1260 FOR J = 12 TO 36 STEP 3: HPLOT 7 * (J - 1) +
     3,I * 2 * 8 TO 7 * (J - 1) + 3,I * 2 * 8 +
     8: NEXT
7C 1270 HPLOT 63,2 * I * 8 + 11 TO 279,2 * I * 8 + 1
     1: NEXT
A6 1280 FOR I = 1 TO A STEP 2: VTAB 20 + (A < 3) + I
     : HTAB 1: PRINT " #"I" "NA$(I)":";: IF NA$(I
     + 1) < > "" THEN HTAB 23: PRINT " #"I + 1"
     "NA$(I + 1)":"
93 1290 NEXT I
AF 1300 HPLOT 0,75 TO 279,75: HPLOT 0,155 TO 279,155
03 1310 FOR I = 0 TO A - 1:S(I) = 1: NEXT : RETURN :
     REM INITIALIZE SCORE STATE
```

Space Dodger

Matthew Marullo
Translation by Rob Terrell and Tim Victor

Try to evade menacing alien ships in this fast, colorful action game. For all Apple II computers using DOS 3.3 or ProDOS.

Get ready for a game which demands extremely sharp hand/eye coordination and judgment of time and distance. "Space Dodger" is an addictive test of your physical reflexes.

Machine Language in Space

Written entirely in machine language, "Space Dodger" works on any Apple II-series computer with any version of Apple DOS. You'll use "AppleMLX" (Appendix C) to enter Space Dodger. After loading and running AppleMLX, you need to type in responses to prompts asking for the beginning and ending addresses of the game. Your answers should be

```
START ADDRESS? 7000
END ADDRESS?   78AF
```

Make sure you've read and understood the instructions for AppleMLX before proceeding. Don't worry if you know nothing about machine language—you don't need to understand it to enter Space Dodger.

When you've finished entering Space Dodger and you've saved it to disk, you'll have a file ready to run. Now type

BRUN ***filename*** (where *filename* is the program's name as it shows on your disk).

The Artful Dodger

There's a brief wait while the program initializes before the game begins. Your spaceship appears on the left side of the screen. On the right side is a lineup of several colorful alien ships. When the action starts, the aliens begin moving toward your ship at different speeds. Your job is to avoid a disastrous midspace collision that will turn your vessel into a lump of smoking metal.

To dodge the reckless aliens, you'll have to move up or down. But don't move too far and try to escape the screen—the boundaries are guarded by cuboids (cube-shaped asteroids)

zipping along at the speed of light. The cuboids are even more dangerous than the alien ships because they travel too fast to dodge.

Press the left-arrow key (←) to move your ship up and the right-arrow key (→) to move down. You have a total of three ships in each game.

Moving Up the Ranks

The longer you evade the oncoming aliens, the more points you gain. However, you won't see your final score until you crash and the game ends. At that time you're ranked according to your value to the Space Service: Space Cadet, Corporal, Sergeant, Captain, or Major.

Every time you advance a rank, the game pauses briefly before it continues to the next level. When it restarts, you'll notice the alien ships fly across the screen even faster. Your score adds up faster, too.

But beware—Space Dodger is not as easy as it looks. Chances are you'll play for quite awhile before you even advance beyond Space Cadet.

Space Dodger

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program.

```
START ADDRESS: 7000
END ADDRESS:   78AF

7000: 4C 14 70 00 21 4C 00 05 59
7008: 40 40 E7 09 00 AA 80 37 89
7010: 44 02 03 05 20 80 76 A9 DD
7018: 20 8D 42 80 A9 40 8D 43 69
7020: 80 AD 57 C0 A9 03 8D 11 29
7028: 70 A9 00 8D 0C 70 8D 0B CC
7030: 70 A9 03 8D 12 70 A9 05 97
7038: 8D 13 70 A9 80 8D 0E 70 14
7040: A9 00 8D 40 80 8D 41 80 E8
7048: 20 C7 71 20 D0 70 20 6A 4E
7050: 70 20 BB 70 20 91 71 B0 CA
7058: E7 20 1B 71 20 E0 75 20 3F
7060: 32 71 20 5D 71 90 E7 4C 7A
7068: F8 71 AD 00 C0 10 2F 2C A8
7070: 10 C0 20 A7 72 C9 95 F0 DE
7078: 14 C9 88 D0 21 AC 07 70 2E
7080: F0 0A 88 8C 07 70 B9 77 1B
7088: 71 8D 05 70 60 AC 07 70 61
```

```
7090: C0 05 F0 0A C8 8C 07 70 C8
7098: B9 77 71 8D 05 70 60 70 56
70A0: B9 77 71 8D 05 70 60 05 F2
70A8: CD 07 70 D0 01 60 EE 07 BB
70B0: 70 AC 07 70 B9 77 71 8D F8
70B8: 05 70 60 A0 05 38 B9 83 4E
70C0: 71 F9 7D 71 B0 03 20 DF 51
70C8: 70 99 83 71 88 10 EE 60 92
70D0: AE 0E 70 A0 05 20 EA 70 93
70D8: 99 83 71 88 10 F7 60 AE ED
70E0: 0D 70 E8 D0 02 8A 60 20 A9
70E8: 95 72 8E 0D 70 C0 05 F0 55
70F0: 21 C0 00 F0 1D AD 0A 70 C5
70F8: 0A 0A 38 6D 0A 70 8D 0A 76
7100: 70 D0 03 AD 12 70 CD 12 8A
7108: 70 F0 0A 90 08 ED 12 70 36
7110: B0 F4 AD 13 70 99 7D 71 C5
7118: A9 80 60 AD 43 80 AC 42 8E
7120: 80 8D 42 80 8C 43 80 29 92
7128: 20 F0 02 A9 01 AA BD 54 B4
7130: C0 60 A0 05 8C 06 70 B9 07
7138: 77 71 8D 86 74 B9 83 71 50
7140: 8D 85 74 F0 04 B9 89 71 74
7148: 2C A9 54 8D 83 74 A9 60 B0
7150: 8D 84 74 20 80 74 AC 06 E0
7158: 70 88 10 D8 60 A9 00 8D 5C
7160: 83 74 A9 60 8D 84 74 AD 72
7168: 04 70 8D 85 74 AD 05 70 48
7170: 8D 86 74 20 80 74 60 07 E9
7178: 14 21 30 3B 48 05 01 03 C2
7180: 01 02 05 1C 47 56 08 0E 78
7188: 1C 38 0E 1C 2A 0E 38 18 1D
7190: 60 A0 05 B9 83 71 D0 F7 83
7198: 88 10 F8 AC 0C 70 C0 04 55
71A0: F0 23 C8 8C 0C 70 AD 12 36
71A8: 70 4A 6D 12 70 8D 12 70 73
71B0: AD 13 70 4A 6D 13 70 8D 08
71B8: 13 70 4E 0E 70 A9 00 8D A3
71C0: 0B 70 20 B0 72 38 60 AD 37
71C8: 50 C0 AD 52 C0 AD 42 80 A0
71D0: C9 20 D0 0D 20 EE 71 AD D8
71D8: 54 C0 20 F3 71 AD 55 C0 07
71E0: 60 20 F3 71 AD 55 C0 20 F5
71E8: EE 71 AD 54 C0 60 A9 20 95
71F0: 4C EA F3 A9 40 4C EA F3 CA
71F8: A9 2C 8D FD 74 20 E0 75 A8
7200: 20 32 71 AD 05 70 8D 86 16
7208: 74 AD 04 70 8D 85 74 A9 2F
7210: 46 8D 83 74 A9 60 8D 84 A1
7218: 74 20 80 74 20 1B 71 A9 90
```

```
7220: 4C 8D FD 74 AD ØD 70 38 50
7228: ED ØE 70 18 6D ØB 70 8D 1D
7230: ØB 70 CE 11 70 F0 03 4C 3B
7238: 40 70 20 58 FC 20 1F 73 FC
7240: AD 10 C0 AD 00 C0 10 FB 12
7248: AD 10 C0 29 5F C9 4E F0 62
7250: 03 4C 24 70 6C FC FF AC 59
7258: 0C 70 B9 6A 72 A8 B9 6F 56
7260: 72 F0 06 20 ED FD C8 D0 47
7268: F5 60 00 06 0F 18 20 C3 9D
7270: C1 C4 C5 D4 00 C3 CF D2 EE
7278: D0 CF D2 C1 CC 00 D3 C5 04
7280: D2 C7 C5 C1 CE D4 00 C3 23
7288: C1 D0 D4 C1 C9 CE 00 CD 90
7290: C1 CA CF D2 00 2C 30 C0 02
7298: 8E 06 70 A2 10 CA D0 FD C9
72A0: 2C 30 C0 AE 06 70 60 A0 FD
72A8: 20 2C 30 C0 88 D0 FA 60 98
72B0: AD 51 C0 AD 54 C0 20 58 F1
72B8: FC A9 05 85 24 A9 06 85 D8
72C0: 25 20 22 FC A9 73 A0 B2 63
72C8: 20 9E 73 A9 05 85 24 A9 9E
72D0: 73 A0 CF 20 9E 73 A9 08 B1
72D8: 85 24 A9 73 A0 EA 20 9E 85
72E0: 73 20 57 72 A9 8D 20 ED 4B
72E8: FD 20 ED FD A9 08 85 24 0F
72F0: A9 73 A0 F7 20 9E 73 A9 27
72F8: F8 A8 8D 0F 70 8D 10 70 71
7300: EE 0F 70 D0 FB EE 10 70 69
7308: D0 F6 C8 D0 F3 60 AD 43 FA
7310: 80 29 40 F0 02 A9 01 A8 F9
7318: B9 54 C0 AD 50 C0 60 AD D7
7320: 51 C0 AD 54 C0 A9 06 85 19
7328: 25 20 22 FC A9 0C 85 24 6A
7330: A9 74 A0 08 20 9E 73 A9 A9
7338: 07 85 24 A9 74 A0 14 20 91
7340: 9E 73 20 57 72 A9 8D 20 42
7348: ED FD A9 07 85 24 A9 74 CF
7350: A0 22 20 9E 73 AD 0C 70 D8
7358: 18 69 B0 20 ED FD AD 0B 8B
7360: 70 A2 03 A0 00 DD 9A 73 B2
7368: 90 06 FD 9A 73 C8 B0 F5 98
7370: 48 98 18 69 B0 20 ED FD 1B
7378: 68 CA D0 E7 A9 8D 20 ED 90
7380: FD 20 ED FD A9 07 85 24 A4
7388: A9 74 A0 31 20 9E 73 A9 94
7390: 05 85 24 A9 74 A0 47 20 4F
7398: 9E 73 60 01 0A 64 8D A8 6D
73A0: 73 8C A7 73 A0 00 B9 00 09
73A8: 00 F0 06 20 ED FD C8 D0 58
```

```
73B0: F5 60 CD C5 D3 D3 C1 C7 F9
73B8: C5 A0 C6 D2 CF CD A0 C8 70
73C0: C5 C1 C4 D1 D5 C1 D2 D4 E0
73C8: C5 D2 D3 BA 8D 8D 00 D9 E9
73D0: CF D5 A0 C8 C1 D6 C5 A0 4B
73D8: C2 C5 C5 CE A0 D0 D2 CF F5
73E0: CD CF D4 C5 C4 A0 D4 CF BB
73E8: 8D 00 D4 C8 C5 A0 D2 C1 D5
73F0: CE CB A0 CF C6 A0 00 C3 BF
73F8: CF CE C7 D2 C1 D4 D5 CC 7B
7400: C1 D4 C9 CF CE D3 A1 00 3E
7408: C7 C1 CD C5 A0 CF D6 C5 13
7410: D2 8D 8D 00 D9 CF D5 D2 04
7418: A0 D2 C1 CE CB A0 C9 D3 73
7420: A0 00 D9 CF D5 D2 A0 D3 A0
7428: C3 CF D2 C5 A0 C9 D3 A0 12
7430: 00 D0 D2 C5 D3 D3 A0 CE 02
7438: A0 D4 CF A0 D1 D5 C9 D4 F8
7440: AC A0 C1 CE D9 8D 00 CF A1
7448: D4 C8 C5 D2 A0 CB C5 D9 4D
7450: A0 D4 CF A0 D0 CC C1 D9 D9
7458: A0 C1 C7 C1 C9 CE 00 D9 7A
7460: A0 C1 C7 C1 A0 A0 00 FF A6
7468: B7 FF 00 FF 00 FF 00 FF 2D
7470: 00 FF 00 FF 00 FF 00 FF 59
7478: 00 FF 00 FF 00 FF 00 FF 61
7480: 4C 8C 74 00 60 21 4C 20 81
7488: 0E 0A 03 03 A9 70 8D 8B 41
7490: 74 A9 00 8D 89 74 AD 85 F5
7498: 74 CD 8B 74 90 04 ED 8B E3
74A0: 74 38 2E 89 74 4E 8B 74 98
74A8: 90 EF C9 04 90 02 69 FB AA
74B0: 2A 8D 8A 74 4A 2E 89 74 3D
74B8: AD 87 74 0A AD 86 74 2A 24
74C0: 8D 88 74 85 A1 AD 83 74 B8
74C8: 85 F8 AD 84 74 85 F9 AD 0C
74D0: 8A 74 0A A8 B1 F8 85 9F 04
74D8: 18 69 02 85 FC C8 B1 F8 28
74E0: 85 A0 85 FD 90 02 E6 FD 9D
74E8: 20 BB 75 20 55 75 A4 9E E9
74F0: B1 FE 11 FC 48 51 FE 09 F2
74F8: 80 D1 FC F0 03 4C D5 75 AF
7500: 68 91 FE 88 10 EA 18 A0 E8
7508: 01 B1 9F 65 FC 85 FC 90 B2
7510: 02 E6 FD E6 A1 C6 9D D0 18
7518: D2 A9 80 85 FF A9 00 2C 12
7520: 42 80 50 02 A9 20 85 FE 4E
7528: 2C 42 80 50 08 AD 41 80 C8
7530: EE 41 80 10 06 AD 40 80 DB
7538: EE 40 80 0A 0A A8 A5 9F 39
```

```
7540: 91 FE C8 A5 A0 91 FE C8 39
7548: AD 88 74 91 FE C8 AD 89 D3
7550: 74 91 FE 18 60 A5 A1 29 41
7558: 3F A8 B9 7B 75 0D 42 80 E0
7560: 85 FF A9 08 25 A1 F0 02 57
7568: A9 80 18 24 A1 10 02 69 48
7570: 50 50 02 69 28 6D 89 74 EC
7578: 85 FE 60 00 04 08 0C 10 5A
7580: 14 18 1C 00 04 08 0C 10 67
7588: 14 18 1C 01 05 09 0D 11 8E
7590: 15 19 1D 01 05 09 0D 11 77
7598: 15 19 1D 02 06 0A 0E 12 9E
75A0: 16 1A 1E 02 06 0A 0E 12 87
75A8: 16 1A 1E 03 07 0B 0F 13 AE
75B0: 17 1B 1F 03 07 0B 0F 13 97
75B8: 17 1B 1F A0 00 B1 9F 85 6F
75C0: 9D C8 B1 9F 38 E9 01 85 CD
75C8: 9E A9 27 ED 89 74 C5 9E 79
75D0: B0 02 85 9E 60 68 38 60 A4
75D8: 00 FF 00 FF 00 FF 00 FF C3
75E0: 4C E5 75 02 18 A9 00 8D 2F
75E8: E4 75 A9 80 85 F9 A9 00 48
75F0: 2C 42 80 50 02 A9 20 85 14
75F8: F8 2C 42 80 50 05 AD 41 EE
7600: 80 10 03 AD 40 80 8D E3 6F
7608: 75 AD E3 75 D0 03 4C 66 80
7610: 76 AC E4 75 B1 F8 85 9F 73
7618: 18 69 02 85 FC C8 B1 F8 6B
7620: 85 A0 85 FD 90 02 E6 FD E0
7628: C8 B1 F8 8D 88 74 85 A1 A0
7630: C8 B1 F8 8D 89 74 C8 8C 22
7638: E4 75 20 BB 75 20 55 75 01
7640: A4 9E B1 FC 49 FF 31 FE D8
7648: 91 FE 88 10 F5 18 A0 01 22
7650: B1 9F 65 FC 85 FC 90 02 BD
7658: E6 FD E6 A1 C6 9D D0 DD 5B
7660: CE E3 75 4C 09 76 A9 00 96
7668: 2C 42 80 50 04 8D 41 80 6A
7670: 60 8D 40 80 60 FF 00 FF 04
7678: 00 FF 00 FF 00 FF 00 FF 65
7680: 4C 87 76 00 03 00 00 A9 06
7688: 07 8D 86 76 A9 00 85 FE EB
7690: A9 60 85 FF A9 FC 85 FC 64
7698: A9 60 85 FD A9 40 85 F8 55
76A0: A9 77 85 F9 A0 00 A5 FC DD
76A8: 91 FE E6 FE A5 FD 91 FE 32
76B0: E6 FE B1 F8 91 FC 8D 83 B5
76B8: 76 C8 B1 F8 91 FC 8D 84 F8
76C0: 76 C8 AE 84 76 B1 F8 91 37
76C8: FC C8 CA D0 F8 CE 83 76 4D
```

```
76D0: D0 F0 18 98 65 F8 85 F8 02
76D8: 90 02 E6 F9 A9 06 8D 85 11
76E0: 76 18 A5 FC 85 FA 98 65 42
76E8: FC 85 FC A5 FD 85 FB 90 3E
76F0: 02 E6 FD A0 00 A5 FC 91 84
76F8: FE E6 FE A5 FD 91 FE E6 74
7700: FE B1 FA 91 FC 8D 83 76 EE
7708: C8 B1 FA 91 FC 8D 84 76 DD
7710: C8 AE 84 76 18 B1 FA 2A AE
7718: 30 03 18 09 80 91 FC C8 80
7720: CA D0 F2 CE 83 76 D0 E9 75
7728: CE 85 76 D0 B4 CE 86 76 20
7730: F0 0D 18 98 65 FC 85 FC 8E
7738: 90 02 E6 FD 4C A4 76 60 EE
7740: 0B 04 80 88 80 80 80 AA FF
7748: 80 80 C0 AA 81 80 D0 AA B4
7750: 85 80 D4 AA 95 80 FE FF 14
7758: BF 80 A8 D5 8A 80 FE FF 0E
7760: BF 80 D0 AA 85 80 D0 AA 8E
7768: 85 80 80 AA 80 80 15 03 28
7770: C0 81 80 F0 87 80 98 8C 3B
7778: 80 CC 99 80 CC 99 80 98 7C
7780: 8C 80 F0 87 80 C0 81 80 F6
7788: C0 81 80 C0 81 80 FE 8F EF
7790: 80 F8 BF 80 C0 81 80 C0 CB
7798: 81 80 F0 87 80 98 8C 80 FE
77A0: CC 99 80 CC 99 80 98 8C C5
77A8: 80 F0 87 80 C0 81 80 0D 27
77B0: 04 C0 BF 80 80 E0 FF 80 D9
77B8: 80 B0 D5 81 80 B8 D5 83 FC
77C0: 80 AC D5 86 80 FE FF 87 C5
77C8: 80 D6 AA 87 80 FE FF 87 03
77D0: 80 AE D5 86 80 AC D5 83 B4
77D8: 80 B8 D5 81 80 F0 FF 80 51
77E0: 80 E0 BF 80 80 0B 05 80 03
77E8: 80 80 90 80 80 80 80 95 EE
77F0: 80 80 FF 83 84 80 C0 AA CA
77F8: AD 91 80 A0 D5 88 84 80 97
7800: 9E 80 AC 81 80 A0 D5 88 C8
7808: 94 80 C0 AA AD 81 80 80 1B
7810: FF 83 95 80 80 80 80 84 28
7818: 80 80 80 80 90 80 08 03 1B
7820: F0 9F 80 98 9C 80 8C 9E A9
7828: 80 FE 99 80 C6 99 80 C6 B8
7830: 8D 80 C6 87 80 FE 81 80 DC
7838: 14 04 80 90 80 80 88 95 F9
7840: 82 80 A8 D5 8A 80 E8 DD 0B
7848: AB 80 A8 F7 AE 80 EA DD EE
7850: AB 80 BA F7 AE 81 EA FD 5D
7858: BB 81 EA FF AE 81 E8 DD 10
```

```
7860: AB 80 A8 FF AF 81 EA DF 95
7868: BB 81 BA FF AF 81 EA FD 46
7870: BB 81 E8 FF AF 81 EA DF F5
7878: AB 80 BA F7 AE 80 EA DD 61
7880: AB 80 AA D5 8A 80 A8 D5 97
7888: 82 80 0A 03 80 80 80 80 D3
7890: 80 80 80 80 80 80 80 80 81
7898: 80 80 80 80 80 80 80 80 89
78A0: 80 80 80 80 80 80 80 80 91
78A8: 80 80 00 00 00 00 00 00 F9
```

3

Education and Logic

Starving Artist

Clark and Kathy H. Kidd

This simple drawing program for young children lets them create almost any picture on an Apple II computer screen. "Starving Artist" works in either DOS 3.3 or ProDOS.

You're dozing through a late-night monster movie, and suddenly you're jolted awake by the shouting of an announcer. Commercial time. And what excitement! There's going to be an art sale at the local inn. You listen carefully as the announcer tells you the news:

Clowns! Seascapes! Animals! You'll find them all at the art fair! Come on down this weekend!

Suddenly you're wide awake. You're a starving artist yourself, and if anyone deserves to exhibit paintings at an art fair, it's you. If you paint some masterpieces tonight, they may even be good enough to sell. You can pay the rent, and somebody will have a work of art to hang over the living room sofa.

How to Play

Before you draw your picture using "Starving Artist," select the background color (or shade) you desire. The numbers will appear on your computer. Type the number of the color you select, and then press the Return key to begin drawing. Press the space bar at any time to change the color of your paint and use these keys to move your brush across the canvas:

W = Up
Z = Down
A = Left
S = Right

You can also move diagonally by pressing different keys in sequence. For example, if your "paintbrush" is on the left edge of your canvas and you want to move toward the upper-right corner, press S, then W, then S, then W, then S again (and so on) until you've gone up as far as you want to go.

If you want to "pick up" your paintbrush and move it to another place on the canvas, just keep hitting the space bar until the brush's color matches the background. Now you can

move it with the W, Z, A, and S keys—the line it draws will be invisible. When you think you're at the right place, press the space bar again to turn your brush into a visible color.

When you've finished one masterpiece, press the Escape (ESC) key to clear the screen so that you can draw another. If you want to end Starving Artist, type the number 99 and then press the Return key instead of choosing a new background color.

Starving Artist

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
11 100 REM STARVING ARTIST
56 110 TEXT : HOME
0F 120 HTAB 13: VTAB 8: PRINT "S T A R V I N G"
03 130 HTAB 15: VTAB 11: PRINT "A R T I S T"
BD 140 HTAB 10: VTAB 23: FLASH : PRINT "PRESS ANY KE
      Y TO START": NORMAL
EB 145 GOSUB 9500
C0 150 X = PEEK ( - 16384): IF X < 128 THEN 150
18 160 POKE - 16368,0
45 200 HOME
0F 210 PRINT : PRINT "THIS GAME TURNS YOUR SCREEN IN
      TO A       CANVAS AND YOUR KEYBOARD INTO AN"
C4 220 PRINT "ARTIST'S BRUSH."
E9 230 PRINT : PRINT "PRESS THE ";: INVERSE : PRINT
      "SPACE";: NORMAL : PRINT " BAR TO CHANGE THE"
AB 240 PRINT "COLOR OF YOUR PAINT."
E1 250 PRINT : PRINT "PRESS THE ";: INVERSE : PRINT
      "ESC";: NORMAL : PRINT " KEY TO RESTART WITH
      A"
21 260 PRINT "FRESH CANVAS."
CB 270 PRINT : PRINT "USE THE FOLLOWING KEYS TO MOVE
       YOUR      BRUSH ACROSS THE CANVAS:"
3F 280 PRINT : PRINT "   ";: INVERSE : PRINT "W";: NO
      RMAL : PRINT " = MOVE UP"
A1 290 PRINT "   ";: INVERSE : PRINT "Z";: NORMAL : P
      RINT " = MOVE DOWN"
0C 300 PRINT "   ";: INVERSE : PRINT "A";: NORMAL : P
      RINT " = MOVE LEFT"
7F 310 PRINT "   ";: INVERSE : PRINT "S";: NORMAL : P
      RINT " = MOVE RIGHT"
E8 320 HTAB 5: VTAB 23: FLASH : PRINT "PRESS ANY KEY
       TO START PAINTING"
7E 330 X = PEEK ( - 16384): IF X < 128 THEN 330
16 340 POKE - 16368,0
D0 350 NORMAL
```

```
9E 1000 REM *** START A NEW SCREEN
1C 1010 TEXT : HOME : PRINT : PRINT "ENTER THE BACKG
      ROUND COLOR YOU DESIRE:": PRINT : PRINT
18 1020 PRINT "    0  - BLACK"
75 1030 PRINT "    1  - MAGENTA"
2A 1040 PRINT "    2  - DARK BLUE"
CE 1050 PRINT "    3  - PURPLE"
22 1060 PRINT "    4  - DARK GREEN"
43 1070 PRINT "    5  - LIGHT GREY"
6C 1080 PRINT "    6  - MEDIUM BLUE"
70 1090 PRINT "    7  - LIGHT BLUE"
83 1100 PRINT "    8  - BROWN"
1E 1110 PRINT "    9  - ORANGE"
21 1120 PRINT "   10  - DARK GREY"
F8 1130 PRINT "   11  - PINK"
32 1140 PRINT "   12  - LIGHT GREEN"
FF 1150 PRINT "   13  - YELLOW"
86 1160 PRINT "   14  - AQUA"
21 1170 PRINT "   15  - WHITE"
37 1180 PRINT "   99  - ";: INVERSE : PRINT "QUIT": N
      ORMAL
80 1200 HTAB 3: VTAB 23: INPUT "";X$
E3 1210 IF LEN (X$) < 1 THEN PRINT CHR$ (7);: GOTO 1
      200
30 1220 Y = 0: FOR X = 1 TO LEN (X$): IF MID$ (X$,X,
      1) < "0" OR MID$ (X$,X,1) > "9" THEN Y = 1:
      NEXT
9E 1230 IF Y = 1 THEN PRINT CHR$ (7);: GOTO 1200
33 1240 BC = VAL (X$)
1E 1250 IF BC = 99 THEN GOSUB 9500: HOME : END
0D 1260 IF BC < 0 OR BC > 15 THEN PRINT CHR$ (7);: G
      OTO 1200
3C 2000 REM *** DRAW BACKGROUND
CC 2010 GR : HOME
AF 2020 POKE - 16302,0
31 2030 COLOR= BC
27 2040 FOR X = 0 TO 47
C5 2050 HLIN 0,39 AT X: NEXT
7D 2060 CC = 15: IF BC = 15 THEN CC = 0
F0 2070 COLOR= CC:C1 = 20:R1 = 24: PLOT C1,R1
C6 3000 REM *** GET NEW BRUSH POSITION
83 3010 X = PEEK ( - 16384): IF X < 128 THEN 3010
F3 3020 X = X - 128: POKE - 16368,0
26 3030 IF X = 27 THEN TEXT : HOME : GOTO 1000
B7 3040 IF X < > 32 THEN 3100
F3 3050 CC = CC + 1: IF CC > 15 THEN CC = 0
42 3060 COLOR= CC
0A 3070 PLOT C1,R1
72 3080 GOTO 3010
16 3100 IF X > 96 AND X < 123 THEN X = X - 32
```

```
F1 3110 C2 = C1:R2 = R1
D9 3120 IF X = 87 THEN R2 = R2 - 1: GOTO 3200
C0 3130 IF X = 90 THEN R2 = R2 + 1: GOTO 3200
F9 3140 IF X = 65 THEN C2 = C2 - 1: GOTO 3200
C0 3150 IF X = 83 THEN C2 = C2 + 1
FE 3200 IF C2 < 0 OR C2 > 39 THEN 3010
36 3210 IF R2 < 0 OR R2 > 47 THEN 3010
6B 3220 R1 = R2:C1 = C2
FD 3230 PLOT C1,R1
66 3240 GOTO 3010
6D 9500 REM SUBROUTINE TO PLAY SONGS
43 9510 RESTORE
F8 9520 READ QX$: IF QX$ < > "$SONG" THEN 9520
65 9530 FOR X = 866 TO 891: READ Y: POKE X,Y: NEXT
35 9540 READ X,Y: IF X < 0 THEN RETURN
A1 9550 POKE 864,Y: POKE 865,X: CALL 866: GOTO 9540
FD 9560 DATA "$SONG",172,97,3,174,97,3,232,208,253,1
        69,4,32,168,252,173,48,192,136,208,239,206,9
        6,3,208,231,96
9B 9570 DATA 177,4,192,4,202,4,202,4,202,4,206,4,202
        ,8,192,4,202,4,192,4,182,4,-1,0
```

Mindbusters

Ned W. Schultz

This graphics puzzle game is both challenging and unusually fascinating. For all Apple II-series computers using either DOS 3.3 or ProDOS.

Are you ready to pit your brain against the computer's? "Mindbusters" presents you with three graphics puzzles that are guaranteed to keep your mind's microprocessors and memory chips whirring for hours.

After you type, save, and run Mindbusters, you can choose to solve one of three puzzles: a mind bender, a mind bruiser, or a mind blower. Warm up with the mind bender—it's the easiest. When you're prepared to press your brain to its limits, you're ready for the mind blower.

Following your selection, the program constructs a puzzle and displays it at the upper-left corner of the screen. Your job is to match that puzzle with the one in the workspace at the lower-right corner of the screen. What's more, you're trying to solve the puzzle in as little time as possible. A timer ticks away as you work. There's no limit to how much time you can take, but the timer lets you compare your progress to a previous performance, or if you want, against another player. Your fastest time during the current session will be displayed on the screen.

Each puzzle is composed of several horizontal rows of odd shapes. An arrow to the right of the workspace points to the row you're currently working on. To work on different rows, you can move the arrow up and down with the I and M keys (make sure you have the Caps Lock key pressed down if you're using Mindbusters on an Apple IIe or IIc). To move the row of shapes beside the arrow left or right, press the J or K key. Keep moving the row left or right until you think you've matched that row to the master puzzle's pattern. Then start working on another row.

When you correctly match all the rows, the program automatically signals that you've solved the puzzle. You can play again if you like.

Helpful Hints

Because Mindbusters can generate a tremendous number of different puzzles, there are very few tricks to mastering it. I suggest you work from top to bottom or vice versa. The best tip I can offer after hours of my own mindbusting is to concentrate, concentrate, concentrate.

Important: When typing in the program, be extra careful with the long strings of characters at the beginning of the listing. These strings become the puzzle shapes. If you mistype or transpose a couple of characters when typing these strings, the program may still run, but it won't know when you've solved the puzzle. Of course, if you're using the "Apple Automatic Proofreader" (Appendix B) to enter the listing, you'll have a lot of help. The Proofreader makes it almost impossible to make mistakes—it even catches character-transposition errors.

ProDOS Changes

Change lines 120 and 450 in the program listing to what you see below:

```
120 POKE 6,0: POKE 7,141: PRINT CHR$ (4);"PR# A$3
    00"
450 PRINT CHR$ (4);"PR#0"
```

Mindbusters

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
08 10 HIMEM: 36096
42 20 R1 = 0:R2 = 0:H$ = "                                "
CA 30 GOSUB 620
16 40 A$ = "'%!&%4&&!'$!&%4&&!'!&%4&!$!$!%%&!$$$'!&%
    &!'&!'#%$%$&!&%4$&##'%$%!&$&"
F1 50 B$ = "/0(.+(+(*.)0+-(-..*+-.0++0+(/*)++0/*+0(+
    ).++0/.(*+(+(/)/00+)+/()+*(+"
3E 60 C$ = "1122211221212121211111212221222121222212
    12111121212221211111212212"
93 70 TEXT : HOME : VTAB 5: PRINT "    ";: FOR I = 4
     TO 35: PRINT CHR$ (95);: NEXT : PRINT
60 80 PRINT : PRINT SPC( 14)"MINDBUSTERS"
D6 90 PRINT "    ";: FOR I = 4 TO 35: PRINT CHR$ (95
    );: NEXT : PRINT : VTAB 11: PRINT TAB( 15)"PLE
    ASE WAIT...": GOSUB 440
01 100 HGR : HCOLOR= 5: HPLOT 1,3 TO 95,3 TO 95,75 T
     O 1,75 TO 1,3
```

```
34 110 HCOLOR= 6: HPLOT 154,76 TO 250,76 TO 250,147
       TO 154,147 TO 154,76
76 120 POKE 6,0: POKE 7,141: POKE 54,0: POKE 55,3: C
       ALL 1002
76 130 FOR N = 1 TO 8:PP(N) = INT ( RND (3) * 55) +
       1: HTAB 2: VTAB 1 + N: PRINT MID$ (D$,PP(N),1
       2): NEXT
99 140 FOR N = 1 TO 8:P(N) = INT ( RND (2) * 55) + 1
       : VTAB 10 + N: HTAB 24: PRINT MID$ (D$,P(N),1
       2): NEXT
F6 150 VTAB 11: HTAB 38: PRINT "3"
CE 160 AC = 1: VTAB 21: PRINT TAB( 14)"USE THE I, J,
        K AND M": PRINT TAB( 14)"KEYS TO MATCH THE P
       ATTERN": PRINT TAB( 14)"IN THE RED BOX AS FAS
       T": PRINT TAB( 14)"AS YOU CAN!!!!";
CE 170 KE = PEEK ( - 16384):J = 0: POKE - 16368,0:J
       = KE - 200
F7 180 T3 = T3 + 1: IF T3 = 12 THEN T3 = 0:T2 = T2 +
        1: IF T2 = 60 THEN T2 = 00:T1 = T1 + 1
06 190 IF J < 0 OR J > 5 THEN J = 0
26 200 ON J GOTO 240,320,300,170,270
2B 210 HTAB 1: VTAB 22: PRINT "RECORD "R1":";: IF R2
        < 10 THEN PRINT "0";
AF 220 PRINT R2: HTAB 1: VTAB 24: PRINT "TIME "T1":"
       ;: IF T2 < 10 THEN PRINT "0";
5E 230 PRINT T2;: GOTO 170
75 240 VTAB 10 + AC: HTAB 38: PRINT " "
C6 250 AC = AC - 1: IF AC < 1 THEN AC = 1
97 260 VTAB 10 + AC: HTAB 38: PRINT "3";: GOTO 210
D7 270 AC = AC + 1: IF AC > 8 THEN AC = 8
C0 280 VTAB 9 + AC: HTAB 38: PRINT " "
24 290 GOTO 260
96 300 P(AC) = P(AC) - 1: IF P(AC) < 1 THEN P(AC) =
       1
92 310 GOTO 330
7E 320 P(AC) = P(AC) + 1: IF P(AC) > 56 THEN P(AC) =
        56
D8 330 VTAB 10 + AC: HTAB 24: PRINT MID$ (D$,P(AC),1
       2)
13 340 FOR X = 1 TO 8: IF PP(X) < > P(X) THEN 210
05 350 NEXT
2A 360 FOR I = 21 TO 23: VTAB I: HTAB 14: PRINT H$:
       NEXT : VTAB 24: HTAB 14: PRINT MID$ (H$,1,14)
       ;: FOR I = 1 TO 5: PRINT CHR$ (7);: NEXT
6B 370 VTAB 22: HTAB 20: PRINT "PUZZLE SOLVED!": HTA
       B 20: PRINT "PLAY AGAIN?"
31 380 HTAB 26: PRINT "Y/N";: GET K$
20 390 IF K$ = "N" THEN TEXT : HOME : END
2E 400 T$ = STR$ (T1) + "." + STR$ (T2):R$ = STR$ (R
       1) + "." + STR$ (R2)
```

```
6A 410 IF R$ = "0.0" OR VAL (T$) < VAL (R$) THEN R1
       = T1:R2 = T2
6B 420 IF K$ = "Y" THEN T1 = 0:T2 = 0:T3 = 0: GOTO 7
       0
9C 430 GOTO 380
D2 440 POKE 230,32: CALL - 3086: POKE 230,64: CALL -
        3086
57 450 POKE 54,240: POKE 55,253: CALL 1002
57 460 FOR P = 1 TO 2: HCOLOR= P: FOR I = 1 TO 8
8B 470 HPLOT I,I TO 279 - I,I TO 279 - I,191 - I TO
       I,191 - I TO I,I
AE 480 NEXT I: POKE 230,32: NEXT P
95 490 VTAB 11: HTAB 26: PRINT "    "
2E 500 VTAB 11: HTAB 11: PRINT "DO YOU WANT TO:": PR
       INT : PRINT TAB( 11)"1 - BEND YOUR MIND?": PR
       INT : PRINT TAB( 11)"2 - BRUISE YOUR MIND?":
       PRINT : PRINT TAB( 11)"3 - BLOW YOUR MIND?"
81 510 POKE - 16302,0
54 520 A = PEEK ( - 16384): IF A > 127 THEN 550
4F 530 POKE - 16297,0: POKE - 16304,0: POKE - 16300,
       0: POKE - 16299,0: POKE - 16300,0: POKE - 163
       03,0: FOR I = 1 TO 50: NEXT
9A 540 GOTO 520
A2 550 POKE - 16368,0:A = A - 176: IF A < 1 OR A > 3
        THEN 520
F1 560 POKE 230,32: CALL - 3086
EC 570 IF A = 1 THEN D$ = A$
EF 580 IF A = 2 THEN D$ = B$
F2 590 IF A = 3 THEN D$ = C$
16 600 RETURN
AA 610 REM SHAPE DATA
FC 620 FOR I = 36096 TO 36263: READ A:CS = CS + A: P
       OKE I,A: NEXT
3B 630 IF CS < > 11534 THEN PRINT "ERROR IN FIRST SE
       T OF DATA STATEMENTS.": STOP
AF 640 DATA 128,128,128,128,128,128,128,128
B1 650 DATA  0,0,0,0,255,255,255,255
5E 660 DATA 0,0,0,0,0,0,0,0
E8 670 DATA 0,0,0,0,0,0,0,255
49 680 DATA 0,0,0,0,0,0,255,255
4B 690 DATA 255,255,0,0,0,0,0,0
86 700 DATA 255,255,255,0,0,0,0,0
C4 710 DATA 0,0,0,0,0,255,255,255
D4 720 DATA 24,24,24,31,31,24,24,24
13 730 DATA 24,24,24,31,31,0,0,0
79 740 DATA 0,0,0,248,248,24,24,24
02 750 DATA 0,0,0,31,31,24,24,24
EE 760 DATA 24,24,24,255,255,0,0,0
2A 770 DATA 0,0,0,255,255,24,24,24
```

```
AC 780 DATA 24,24,24,248,248,24,24,24
9F 790 DATA 24,24,24,248,248,0,0,0
F2 800 DATA 24,24,24,255,255,24,24,24
D6 810 DATA 204,153,51,102,204,153,51,102
54 820 DATA 51,153,204,102,51,153,204,102
57 830 DATA 8,12,14,127,127,14,12,8
F3 840 DATA  255,0,0,0,0,0,0,0
44 850 REM HROUT ML ROUTINE
5D 860 FOR I = 768 TO 856: READ A:CK = CK + A: POKE
       I,A: NEXT
E5 870 IF CK < > 8413 THEN PRINT "ERROR IN SECOND SE
       T OF DATA STATEMENTS.": STOP
28 880 RETURN
D1 890 DATA 216,120,133,69,134,70,132,71
59 900 DATA 166,7,10,10,176,4,16,62
9C 910 DATA 48,4,16,1,232,232,10,134
AA 920 DATA 27,24,101,6,133,26,144,2
A7 930 DATA 230,27,165,40,133,8,165,41
23 940 DATA 41,3,5,230,133,9,162,8
57 950 DATA 160,0,177,26,36,50,48,2
E3 960 DATA 73,127,164,36,145,8,230,26
FA 970 DATA 208,2,230,27,165,9,24,105
60 980 DATA 4,133,9,202,208,226,165,69
64 990 DATA 166,70,164,71,88,76,240,253
9F 1000 DATA 255,255,255,255,255,255,255,255
```

Chess

John Krause

This impressive chess game is not only fast, but includes five levels of play. Play the computer, let the computer play itself, or use the game to solve difficult chess problems. For Apple II+, IIe, and IIc computers with at least 48K RAM using either DOS 3.3 or ProDOS.

The world was amazed, in the late eighteenth century, by a machine that had the astonishing ability to play a good game of chess. It entertained kings and queens. It defeated Napoleon, a master tactician. Hundreds of people paid to compete against it. Eventually, however, it was revealed that a small man was hidden inside the machine.

A chess-playing machine remained only a dream until the late 1950s, when the first computer chess game was played. Now, the World Computer Championship, held every three years since 1974, attracts almost as much publicity as the human championship matches. Why has there been so much interest in machines that play games?

One reason is that chess can be used to measure a computer's intelligence. Chess is easy to play, but difficult to master. So difficult, in fact, that some experts believe that a computer would have to be almost as intelligent as a human to become world champion.

Of course, another reason is that chess is just plain fun—but not if you can't find an opponent. To be an entertaining opponent, a computer chess game should be fast, easy to use, and capable of playing at several different skill levels. "Chess" has all these features, and more. Although it's really no match against the best commercial chess games, it *has* managed to defeat these giants of the microcomputer chess world on rare occasions.

Typing It In

Chess is a machine language program. That's what gives it its power and speed. Fortunately, you don't have to know machine language to type it in. The program uses BASIC DATA statements to load the machine language into your Apple's

memory. If you've entered a BASIC program before, you won't have any trouble with Chess. However, you should have a copy of "Apple Automatic Proofreader" on disk before beginning. Make sure you read Appendix B and have this program available.

Once you've typed in Chess, save it to disk. *Do not run the program before saving it.* If you do, and you've made a typing mistake, there's a chance the program will lock up the computer. The keyboard won't respond, and you'll have to turn the computer off and on to regain control. All your work will have been lost.

Load and run Chess and you're ready.

Keyboard Input

After running the program, you'll be asked to specify several play options. You can choose among five skill levels, start a new game or set up any position, play against the computer or watch it play against itself, or play with either the white or black pieces. All of these options will be discussed in greater detail later, but for now, just type 1 at each prompt. This puts you in command of the white pieces versus the computer on level 1, the easiest level.

The first time the program is run, you need to wait a few seconds while the computer gets its brain in order. Then the board will be displayed with your pieces on the bottom of the screen and the computer's pieces on the top. The square in the lower-left corner of the board should be blinking. This is the cursor which takes the place of your hand to move pieces around the board.

If you have an Apple IIe or IIc, press down the Caps Lock key. Use the A, D, W, and S keys to move the blinking square left, right, up, or down. Place the cursor square atop the piece you wish to move and press Return. You'll hear a beep. Now maneuver the cursor to the square you want to move the piece to and hit Return again. Your piece appears in the new position and the computer responds almost instantly with its move.

A Spectacular Blunder

Did you make a foolish move? No problem. One of the most valuable features of Chess is its ability to change piece positions by adding or deleting pieces. This feature is especially useful for those of us who frequently manage to maneuver

into a superior position, only to throw it all away in a single, spectacular blunder.

A piece can be deleted by positioning the cursor on it and pressing the space bar. To add a piece or change a piece to a different one, move the cursor to the appropriate square and press P, N, B, R, Q, or K for pawn, knight, bishop, rook, queen, or king, respectively. This puts one of *your* pieces on the square. To add one of the computer's pieces, hold down the Control key while pressing one of these editing keys.

To take back a move, use the editing keys to delete your piece and put it back on its original square. Don't forget to take back the computer's move, too.

The editing feature also enables you to make special moves such as castling and *en passant* captures. For example, castling can be accomplished by deleting the king and putting it on its new square, and then moving the rook as you normally would. Although *you* can make these special moves, the computer will never castle or capture *en passant* because, due to their complexity, these moves were not included in its thinking routine.

Strange Chess

Although the computer always makes a legal move, it doesn't check to see that you do the same. You're free to move any of your pieces to any square without so much as a contemptuous blap from the computer. If you're an experienced player, this shouldn't be a problem. If you're a beginner, however, you may want to familiarize yourself with the basic rules of chess lest you end up playing strange chess, a personal version which bears little resemblance to the real thing. On the other hand, if you like to fudge a bit, the computer makes it easy. It politely acquiesces to your most surreal moves.

When a pawn reaches the other side of the board, it's automatically promoted to a queen. If you would rather have a knight, bishop, or rook, you can easily make the change using the editing keys.

Checkmate

The computer thinks by analyzing thousands of possible moves and countermoves and choosing what it considers to be the best move based on the relative value of the pieces (see "How Chess Thinks" below). Most positions don't have just

one best move, but several which are equally good, in which case the computer chooses among them at random. This random factor insures that every game is different, and makes for varied and interesting play.

Play continues until one side is either checkmated or stalemated. The computer then stops play and indicates which side has won.

There are a few quirks in the way the computer determines whether checkmate has occurred. On levels 3–5, it announces checkmate prematurely. When this happens, the computer has determined that it's impossible to avoid checkmate on the *next* move or two, assuming both sides make the best moves.

Also, the computer doesn't know the subtle difference between checkmate and stalemate. Consequently, when stalemate occurs, it announces checkmate although, in fact, the game is a draw. Since the computer tries as hard as it can to checkmate its opponent, it also tries to achieve stalemate, possibly forcing a draw when it could have won. Fortunately, this rarely happens because the conditions for stalemate exist only in unusual circumstances such as when one side has only the king remaining.

Also, the computer won't give you any hint when your king is in check (not checkmate). So be extra careful that you don't leave your king in check or move into check. Otherwise, your king would be in check during the computer's turn to move—a highly unorthodox, if not illegal, position. The computer's reply to such a position is unpredictable, but it usually announces checkmate, forcing you to restart the game.

In any case, when the computer announces checkmate, press any key to start a new game. If you want to try out some of the other play options without waiting until checkmate, you can start a new game at any time by pressing Control-Reset and rerunning the program.

Play Options

When you choose the black pieces, the board revolves so that you're still playing from the bottom. Since the player with the white pieces always moves first, you must wait for the computer to move before you can make your first move.

If you become mentally exhausted after several bouts against the computer, give your brain a rest and watch the

computer play itself. When you select this option, just sit back and watch the action. Beginners will find this feature an excellent way to learn some good strategies to use against the computer.

You don't have to begin a game from the starting position. If you choose the option to set up a position, an empty board is displayed—you can place pieces in any position with the editing keys. When the position is set up, the computer starts thinking after you make your first move.

This feature is especially useful for continuing a previous game or creating a problem for the computer to solve. It also allows you to experiment with hypothetical or downright ridiculous positions. Live out your fantasy by giving yourself ten queens versus the computer's lone king. The position doesn't even have to be a legal one. You could invent your own type of chess by giving each side two kings, for example, although the computer may get confused trying to determine when checkmate has occurred.

One of the advantages of a computer opponent is that you can tell the machine exactly how hard you want it to try to beat you. It obediently plays at that level of difficulty. This is important, because it's no fun if you always lose or always win effortlessly.

You have five skill levels to choose from. The difference between one level and another is the number of moves ahead that the computer looks. On level 1, for example, it looks two moves ahead (its move and your reply). Each succeeding level looks ahead one move more than the previous level.

Alas, the smarter play on the higher levels doesn't come without a price. The further ahead the computer looks, the more moves it must examine and, hence, the longer it thinks. The thinking time varies greatly depending on the level (about one second per move on level 1; about two *hours* on level 5).

Here's a rundown of the five levels:

Level 1: Beginner. Thinking time: one second. Look ahead: two moves. Fast but dumb.

Level 2: Intermediate. Thinking time: five seconds. Look ahead: three moves. Provides a reasonable challenge for impatient players.

Level 3: Tournament. Thinking time: two minutes. Look ahead: four moves. Since the usual time limit for tournament

play is 40 moves in two hours, an average of three minutes per move, this level is best suited for serious players.

Level 4: Mate in two. Thinking time: 20 minutes. Look ahead: five moves. Capable of solving most mate-in-two problems.

Level 5: Postal chess. Thinking time: two hours. Look ahead: six moves. Simulates postal chess games where there is no time limit. Can avoid checkmate in two moves.

The thinking times given here are average times. The actual time varies greatly depending on the position. For example, level 5 takes only five seconds with just two kings on the board.

Level 4 can be used to solve mate-in-two problems such as those published in many newspapers. Just select the following options: level 4, setup position, and computer versus itself. Enter the positions using the editing keys, and then make a do-nothing move by positioning the cursor over a white piece and pressing the Return key twice. After several minutes of deep thought, the computer should respond by moving one of the white pieces (the solution) and announcing checkmate. The only mate-in-two problems that the computer cannot solve are those which involve castling, *en passant* captures, or pawn promotion.

How Chess Thinks

You've probably heard that if a monkey sat down at a typewriter and pecked randomly at the keys for long enough, it would eventually type the complete works of Shakespeare. Theoretically, that's possible—given enough time. There's the rub. At a brisk typing speed of 50 words per minute, it would take that poor monkey billions of years just to type "To be, or not to be." Nevertheless, there is power in trial and error.

The Minimax Algorithm

Substitute the monkey for a high-speed computer, and this technique becomes a practical method of imitating intelligence. In fact, it has been used with great success in the field of artificial intelligence. This program uses a popular trial-and-error technique known as the minimax algorithm.

The computer looks at the present board position and mentally moves the pieces through all the possible combinations of future moves and countermoves up to a certain point, say, three moves ahead. For each combination, it calculates a score based on which pieces were captured during the combination. Each piece is worth a certain number of points depending on its general importance: 1 point for a pawn, 3 for a knight or bishop, 5 for a rook, 9 for a queen, and 46 for a king. (Of course, since you lose the game if your king cannot escape capture, the value of a king is actually infinite, but 46 is high enough to make the computer protect the king at all costs.)

When, in a move being examined, the computer captures an opponent's piece, the value of that piece is added to the score. Conversely, when one of the computer's pieces is captured, its value is subtracted from the score. Thus, a high score is considered good for the computer, a low score good for its opponent.

The task is to find the combination that represents the best play for both sides. This combination is not necessarily the one with the maximum score, because while the computer is trying to maximize the score, its opponent is

trying just as hard to minimize it. The best combination gives maximum scores during the computer's moves, minimum scores during the opponent's moves.

After the best combination has been found, the computer's best move in the present position is simply the first move in the combination. The problem has been reduced from analyzing a chess position to finding the maximum and minimum of a series of numbers, which is much better suited to a computer.

50 Million Combinations on Level 5

Like most algorithms based on trial and error, this one requires sifting through an enormous number of combinations to find the best one. Fortunately, a few tricks can be used to reduce the combinations to a manageable number. This algorithm uses a technique called alpha-beta cutoff. It makes the computer search more intelligently, giving it the seemingly paradoxical ability to find the best move without looking at all the possible combinations. On level 5, for example, instead of having to search through roughly 2 billion combinations, it looks at only 50 million.

Even so, it would take BASIC months to generate that many combinations. That's why the algorithm is programmed in machine language. An advanced programming technique known as *recursion* (making a subroutine call itself) is used to generate all the possible combinations of moves. Capable of analyzing about 5000 combinations per second, this routine provides a significant challenge at a reasonable playing speed.

Chess

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
0C 10 HIMEM: 15448
DD 20 HOME : PRINT TAB( 18)"CHESS"
2C 30 PRINT : PRINT TAB( 15)"JOHN KRAUSE"
0B 40 DIM A(12),C(69)
33 50 FOR I = 16256 TO 16263: POKE I,192: NEXT I
9A 60 FOR I = 16264 TO 16383: POKE I,7: NEXT I
```

```
1C 70 FOR I = 16285 TO 16362: READ J: POKE I,J: NEXT
      I
76 80 FOR I = 0 TO 12: READ A(I): NEXT I
E1 90 B(0) = 17118:B(1) = 18142:B(2) = 19166:B(3) =
     20190:B(4) = 21214
88 100 FOR I = 0 TO 69: READ C(I): NEXT I: GOSUB 430
41 110 IF PEEK (16200) < > 96 THEN GOSUB 370
31 120 IF B$ = "2" THEN POKE 16288,6: POKE 16289,5:
     POKE 16358,250: POKE 16359,251
77 130 IF E$ = "1" THEN 150
6B 140 FOR R = 0 TO 7: FOR C = 0 TO 7: POKE 16285 +
     10 * R + C,0: NEXT C: NEXT R
E5 150 HGR2 : FOR R = 0 TO 7: FOR C = 0 TO 7
8B 160 I = PEEK (16285 + 10 * R + C)
51 170 GOSUB 820
FA 180 NEXT C: NEXT R:R = 0:C = 0
05 190 IF A$ = "1" AND B$ = "1" THEN 230
6B 200 IF E$ = "2" THEN GOSUB 540
12 210 GOTO 240
77 220 IF A$ = "2" THEN 240
5A 230 GOSUB 540: POKE 16202,0
86 240 CALL 15486: IF PEEK (16256) < 229 AND PEEK (1
     6256) > 150 THEN 310
39 250 J = PEEK (16252) + 16264:R = INT (J / 10 - 16
     28.5):C = J - 16285 - 10 * R
2F 260 CALL - 198:K = PEEK (J):I = 0: GOSUB 820:I =
     K
7D 270 J = PEEK (16253) + 16264:R = INT (J / 10 - 16
     28.5):C = J - 16285 - 10 * R
54 280 GOSUB 820
53 290 IF PEEK (16256) > 99 OR PEEK (16256) < 28 THE
     N 220
C0 300 Z = 1
F6 310 IF PEEK (16202) THEN Z = Z + 1
DD 320 FOR I = 1 TO 5: CALL - 198: NEXT I
5A 330 K = 2:Z = Z + VAL (B$): IF Z / 2 - INT (Z / 2
     ) THEN L = 15
F6 340 GOSUB 910: GOSUB 900
24 350 IF PEEK ( - 16384) < 128 THEN 350
4E 360 TEXT : RUN
DF 370 PRINT : PRINT : PRINT "PLEASE WAIT ..."
51 380 FOR I = 24576 TO 25275: READ J: POKE I,J:K =
     K + J: NEXT I
FE 390 FOR I = 25276 TO 25339: POKE I,255: NEXT I
93 400 FOR I = 15449 TO 16200: READ J: POKE I,J:K =
     K + J: NEXT I
CD 410 IF K = 134648 THEN RETURN
C1 420 POKE 16200,0: PRINT : PRINT "CHECK DATA STATE
     MENTS": STOP
```

```
86 430 PRINT : PRINT : PRINT "ENTER SKILL LEVEL (1-5
       )";
47 440 GET A$: IF VAL (A$) = 0 OR VAL (A$) > 5 THEN
       440
1F 450 POKE 16201, VAL (A$)
5F 460 PRINT : PRINT : PRINT "(1) NEW GAME OR (2) SE
       T UP POSITION?";
CF 470 GET E$: IF VAL (E$) = 0 OR VAL (E$) > 2 THEN
       470
88 480 PRINT : PRINT : PRINT "COMPUTER VS. (1) YOU O
       R (2) ITSELF?";
E0 490 GET A$: IF VAL (A$) = 0 OR VAL (A$) > 2 THEN
       490
8A 500 POKE 16202,0:B$ = "2": IF A$ = "2" THEN POKE
       16202,16:B$ = "1": RETURN
40 510 PRINT : PRINT : PRINT "YOU HAVE THE (1) WHITE
        OR (2) BLACK     PIECES?";
30 520 GET B$: IF VAL (B$) = 0 OR VAL (B$) > 2 THEN
       520
1B 530 RETURN
0A 540 F = 0
82 550 I = PEEK ( - 16384): POKE - 16368,0
98 560 IF I = 215 AND R < 7 THEN R = R + 1: GOTO 670
85 570 IF I = 193 AND C > 0 THEN C = C - 1: GOTO 670
92 580 IF I = 211 AND R > 0 THEN R = R - 1: GOTO 670
91 590 IF I = 196 AND C < 7 THEN C = C + 1: GOTO 670
E6 600 IF I < 128 OR I = 141 OR F THEN 670
25 610 J = 0
BF 620 IF A(J) = I THEN 650
CF 630 J = J + 1: IF J < 13 THEN 620
9E 640 GOTO 550
5F 650 I = J: IF I > 6 THEN I = 262 - I
9F 660 GOSUB 820: GOTO 540
8D 670 POKE 251,R: POKE 252,C
FD 680 J = 16285 + 10 * R + C:K = PEEK (J)
0A 690 IF I = 141 THEN 740
C2 700 POKE 8,7: CALL 24576
DA 710 FOR J = 0 TO 30: NEXT J
69 720 I = K: GOSUB 850
44 730 FOR J = 0 TO 60: NEXT J: GOTO 550
2C 740 IF F THEN 790
2E 750 IF K = 0 OR K > 6 THEN 550
8E 760 F = 1:R1 = R:C1 = C: CALL - 198
5F 770 IF PEEK ( - 16368) = 141 THEN 770
A7 780 GOTO 550
DF 790 R2 = R:C2 = C:R = R1:C = C1:I = 0
A1 800 K = PEEK (16285 + 10 * R + C): GOSUB 820
D3 810 R = R2:C = C2:I = K
7A 820 IF R = 0 AND I = 255 THEN I = 251
83 830 IF R = 7 AND I = 1 THEN I = 5
```

```
86 840 POKE 16285 + 10 * R + C,I
AE 850 IF I > 6 THEN I = 384 - I
24 860 IF B$ = "1" OR I = 0 THEN 890
74 870 IF I > 6 THEN I = I - 256
38 880 I = I + 128
7C 890 POKE 251,R: POKE 252,C: POKE 8,I: CALL 24576:
       RETURN
99 900 K = 7:M = 3:L = 30
9F 910 FOR J = 0 TO K: FOR I = 0 TO 4: POKE B(I) + M
       + J,C(L):L = L + 1: NEXT I: NEXT J: RETURN
EF 920 DATA 4,2,3,5,6,3,2,4,7,7,1,1,1,1,1,1,1,1,7
06 930 DATA 7,0,0,0,0,0,0,0,0,7,7,0,0,0,0,0,0,0,0,7
08 940 DATA 7,0,0,0,0,0,0,0,0,0,7,7,0,0,0,0,0,0,0,7
2D 950 DATA 7,255,255,255,255,255,255,255,255,7
B5 960 DATA 7,252,254,253,251,250,253,254,252
CD 970 DATA 160,208,206,194,210,209,203,144,142,130,
       146,145,139
57 980 DATA 19,21,19,21,115,68,42,46,42,74,21,20,12,
       20,21
72 990 DATA 85,85,119,87,85,100,68,68,68,68,29,4,12,
       4,28
D9 1000 DATA 72,40,72,8,104,1,64,64,65,0,43,40,56,40
       ,43,103,17,19,17,103
18 1010 DATA 42,106,102,42,42,73,21,29,21,21,59,9,25
       ,9,57,35,37,37,5,35
D9 1020 DATA 165,251,69,252,41,1,133,48
6F 1030 DATA 32,19,96,166,8,208,1,96
AC 1040 DATA 232,134,48,165,48,41,15,168
05 1050 DATA 185,170,96,133,6,185,179,96
ED 1060 DATA 133,7,169,0,133,9,164,251
70 1070 DATA 185,162,96,133,254,165,252,10
06 1080 DATA 10,24,121,154,96,133,253,32
EC 1090 DATA 75,96,165,253,24,105,128,133
50 1100 DATA 253,165,254,56,233,32,133,254
04 1110 DATA 76,75,96,32,90,96,165,254
97 1120 DATA 24,105,4,133,254,201,96,48
F2 1130 DATA 242,96,169,3,133,25,164,48
E7 1140 DATA 240,41,136,240,38,16,19,164
93 1150 DATA 9,177,6,230,9,73,255,164
66 1160 DATA 25,49,253,145,253,198,25,16
14 1170 DATA 238,96,164,9,177,6,230,9
A2 1180 DATA 164,25,17,253,145,253,198,25
BC 1190 DATA 16,240,96,164,9,177,6,230
AB 1200 DATA 9,164,25,145,253,198,25,16
9E 1210 DATA 242,96,84,84,44,44,44,44
13 1220 DATA 4,4,65,64,67,66,65,64
3A 1230 DATA 67,66,188,252,60,124,188,252
54 1240 DATA 60,124,188,96,96,97,97,97
A6 1250 DATA 97,98,98,98
```

```
4A 1260 DATA 0,0,0,0,213,170,213,170,213,170,213,170
        ,213,170,213,170
B6 1270 DATA 213,170,213,170,213,170,213,170,213,170
        ,213,170,213,170,213,170
BA 1280 DATA 213,170,213,170,213,170,213,170,213,170
        ,213,170,213,170,213,170
BE 1290 DATA 213,170,213,170,213,170,213,170,213,170
        ,213,170,213,170,213,170
B2 1300 DATA 0,0,0,0,42,85,42,84,42,85,42,84,42,85,4
        2,84
76 1310 DATA 42,85,42,84,42,85,42,84,42,85,42,84,42,
        85,42,84
7A 1320 DATA 42,85,42,84,42,85,42,84,42,85,42,84,42,
        85,42,84
7E 1330 DATA 42,85,42,84,42,85,42,84,42,85,42,84,42,
        85,42,84
40 1340 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,3,96,0
01 1350 DATA 0,15,120,0,0,15,120,0,0,3,96,0,0,15,120
        ,0
FE 1360 DATA 0,3,96,0,0,3,96,0,0,15,120,0,0,63,126,0
B1 1370 DATA 0,63,126,0,0,0,0,0,0,0,0,0,0,0,0,0
F4 1380 DATA 0,0,0,0,0,0,0,0,0,0,0,1,64,0,0,7,64
46 1390 DATA 0,0,127,64,0,15,127,112,0,63,126,48,0,6
        3,127,112
63 1400 DATA 1,127,127,112,1,127,127,124,7,127,103,1
        24,7,127,96,48
C0 1410 DATA 7,127,120,0,7,127,126,0,7,127,127,64,0,
        0,0,0
64 1420 DATA 0,0,0,0,0,0,0,0,0,60,30,0,0,60,30,0
9C 1430 DATA 1,124,127,64,1,115,127,64,1,79,127,64,1
        ,79,127,64
A6 1440 DATA 0,63,126,0,0,48,6,0,0,63,126,0,0,48,6,0
2A 1450 DATA 7,127,127,112,31,124,31,124,24,0,0,12,0
        ,0,0,0
0A 1460 DATA 0,0,0,0,0,0,0,0,3,103,115,96,3,103,115,
        96
B7 1470 DATA 3,127,127,96,0,96,3,0,0,127,127,0,0,127
        ,127,0
29 1480 DATA 0,127,127,0,0,127,127,0,0,127,127,0,0,9
        6,3,0
42 1490 DATA 3,127,127,96,15,127,127,120,15,127,127,
        120,0,0,0,0
DB 1500 DATA 0,0,0,0,0,0,0,0,0,0,48,24,0,0,48,24,0
10 1510 DATA 96,48,24,12,97,124,126,12,25,124,126,48
        ,25,124,126,48
A8 1520 DATA 31,127,127,112,6,0,1,64,7,127,127,64,7,
        124,127,64
80 1530 DATA 7,127,127,64,6,0,1,64,7,127,127,64,0,0,
        0,0
```

```
65 1540 DATA 0,0,0,0,0,63,120,0,0,51,24,0,30,60,121,
        112
AF 1550 DATA 127,115,31,124,127,127,127,124,127,112,
        31,124,31,124,127,112
B8 1560 DATA 31,127,127,112,6,0,1,64,7,127,127,64,7,
        124,127,64
90 1570 DATA 7,127,127,64,6,0,1,64,7,127,127,64,0,0,
        0,0
B1 1580 DATA 21,12,248,237,235,244,8,19,10,11,1,247,
        246,245,255
7C 1590 DATA 9,11,247,245,9,10,1,246,255,46,9,5,3,3,
        1
A1 1600 DATA 0,1,3,3,5,9,46,120,169,192,141,128,63,1
        62,0
A9 1610 DATA 142,127,63,202,142,126,63,76,97,61,189,
        108,63,24,125
78 1620 DATA 116,63,72,168,185,136,63,188,108,63,153
        ,136,63,104,168
29 1630 DATA 189,76,63,153,136,63,24,105,6,168,174,7
        3,63,169,0
4C 1640 DATA 157,129,63,174,126,63,185,113,60,56,253
        ,129,63,168,169
48 1650 DATA 192,157,129,63,152,224,0,208,34,221,128
        ,63,48,28,208
6A 1660 DATA 11,173,35,192,205,127,63,144,18,141,127
        ,63,140,128,63
6B 1670 DATA 173,108,63,141,124,63,173,116,63,141,12
        5,63,96,221,128
44 1680 DATA 63,48,250,240,248,152,157,128,63,189,75
        ,63,24,105,6
B4 1690 DATA 168,185,113,60,56,253,128,63,221,127,63
        ,48,59,224,1
B9 1700 DATA 240,221,221,127,63,240,50,96,189,108,63
        ,24,125,116,63
6E 1710 DATA 141,75,63,168,185,136,63,172,74,63,208,
        6,201,1,16
CA 1720 DATA 192,48,8,201,0,48,186,201,7,240,182,157
        ,76,63,201
84 1730 DATA 6,240,4,201,250,208,12,169,46,157,128,6
        3,104,104,104
13 1740 DATA 104,76,229,61,188,108,63,185,136,63,172
        ,75,63,153,136
D3 1750 DATA 63,188,108,63,169,0,153,136,63,236,73,6
        3,208,3,76
18 1760 DATA 144,60,232,142,126,63,169,20,157,108,63
        ,169,16,56,237
49 1770 DATA 74,63,141,74,63,254,108,63,188,108,63,1
        85,136,63,201
1C 1780 DATA 7,240,86,172,74,63,240,4,201,0,16,77,19
        2,0,208
```

```
C9 1790 DATA 4,201,1,48,69,201,0,16,9,188,108,63,169
      ,0,56
3D 1800 DATA 249,136,63,201,1,208,6,32,5,62,76,222,6
      1,201,2
34 1810 DATA 208,6,32,192,62,76,222,61,201,3,208,6,3
      2,218,62
B0 1820 DATA 76,222,61,201,4,208,6,32,230,62,76,222,
      61,201,5
75 1830 DATA 208,6,32,242,62,76,222,61,32,47,63,76,2
      22,61,189
4C 1840 DATA 108,63,201,98,48,150,224,0,240,16,169,1
      6,56,237,74
7A 1850 DATA 63,141,74,63,202,142,126,63,76,144,60,1
      73,124,63,24
F4 1860 DATA 109,125,63,141,125,63,88,96,173,74,63,2
      08,89,189,108
F7 1870 DATA 63,24,105,10,168,185,136,63,208,36,169,
      10,157,116,63
51 1880 DATA 32,21,61,189,108,63,201,31,48,21,201,39
      ,16,17,24
FF 1890 DATA 105,20,168,185,136,63,208,8,169,20,157,
      116,63,32,21
EB 1900 DATA 61,189,108,63,24,105,9,168,185,136,63,1
      6,8,169,9
56 1910 DATA 157,116,63,32,21,61,189,108,63,24,105,1
      1,168,185,136
54 1920 DATA 63,16,8,169,11,157,116,63,32,21,61,96,1
      89,108,63
03 1930 DATA 56,233,10,168,185,136,63,208,36,169,246
      ,157,116,63,32
C0 1940 DATA 21,61,189,108,63,201,81,48,21,201,89,16
      ,17,56,233
54 1950 DATA 20,168,185,136,63,208,8,169,236,157,116
      ,63,32,21,61
9D 1960 DATA 189,108,63,56,233,9,168,169,0,217,136,6
      3,16,8,169
EF 1970 DATA 247,157,116,63,32,21,61,189,108,63,56,2
      33,11,168,169
D5 1980 DATA 0,217,136,63,16,8,169,245,157,116,63,32
      ,21,61,96
A3 1990 DATA 169,0,157,84,63,168,185,89,60,157,116,6
      3,32,21,61
91 2000 DATA 254,84,63,188,84,63,192,8,48,237,96,169
      ,4,157,100
AF 2010 DATA 63,169,0,157,84,63,240,22,169,8,157,100
      ,63,169,4
F2 2020 DATA 157,84,63,208,10,169,8,157,100,63,169,0
      ,157,84,63
E9 2030 DATA 168,185,105,60,157,116,63,157,92,63,32,
      21,61,189,108
```

```
88 2040 DATA 63,24,125,116,63,168,185,136,63,208,13,
        189,116,63,24
B7 2050 DATA 125,92,63,157,116,63,76,6,63,254,84,63,
        189,84,63
94 2060 DATA 221,100,63,48,206,96,169,0,157,84,63,16
        8,185,97,60
CD 2070 DATA 157,116,63,32,21,61,254,84,63,188,84,63
        ,192,8,48
E7 2080 DATA 237,96
```

Missile Math

Gerry S. Wick
Translation by Kevin Martin

Educational programs are usually designed to reward correct answers. "Missile Math" does this, but also gives extra points for speed. Here's an entertaining way for young students to learn their math. Works on all Apple II computers using either DOS 3.3 or ProDOS. Paddle controller needed.

Combining school drills and arcade action is probably one of the best ways to get children's attention, and certainly one of the most effective educational ways to use a computer. Flash cards, the traditional way of teaching speed and accuracy in math drills, get tiresome after awhile. But by making the concept entertaining, putting it into a game-like environment, children end up spending more time with math practice. That's exactly what "Missile Math" does. It's fun to play and teaches at the same time.

Missile Math starts with a keyboard-controlled menu on the screen. You have a choice of addition and subtraction or multiplication and division at either slow or fast speeds. Use the space bar to select the menu, then press the Return key to start.

Different Difficulty Levels

After you hit Return, you'll see an instruction screen which tells you, among other things, that you're entering level 1. As you advance from one level to the next, the math problems become more and more difficult.

The problem appears at the bottom of the screen, at the far left. Also near the bottom are five possible answers, together with a missile gun which you control with paddle 0. The object is to position the gun over the correct answer and launch a missile (by pressing the paddle controller's fire button) so that it destroys the enemy spaceship moving across the screen. At the bottom right, you'll see the number of remaining missile guns (you begin with three) and your score.

The addition and subtraction problems are combined, as are multiplication and division. You might see **3+4=?** or

3+?=7 or ?+4=7 when you're playing the addition/subtraction game. All use the + sign, but you actually have to subtract one number from another in the second and third types of problems. It's the same in multiplication and division: The problems all use the multiplication sign (*) but when you see 4*?=16, what you really have to do is divide 4 into 16.

The correct answer randomly appears in one of the five possible places, so you never know in advance where to put the gun. The incorrect answers are chosen so that the correct answer isn't obvious. This discourages guessing.

A Feisty UFO

The UFO moves across the screen at three different heights. The first is near the top of the screen and, on the two successive passes, the UFO moves closer to the bottom. Of course, if you destroy the UFO on the first pass, it doesn't appear at the lower altitudes. Instead, a new problem appears, and the UFO starts again at the highest position.

It's difficult to destroy the UFO at the highest altitude because there's less time to calculate the right answer and fire the missile at the right time. Destroying the UFO at the middle altitude is easier, and easiest yet at the lowest altitude. That's why you get 25 points for a correct answer/hit on the first pass, 10 points for the second, and only 5 points for the third. When the gun is positioned over the correct answer and you score a hit, the UFO turns red, the screen goes black, and another problem appears. Your new score is updated and displayed in the bottom right of the screen.

If you score a direct hit on the UFO, but the gun's over a wrong answer, the UFO briefly changes to white and continues on its way. You have to try again on the next pass. Three consecutive misses or incorrect answers, and the UFO destroys the gun. An encouraging message flies onto the screen and the correct answer zips on. The game ends when all three guns have been destroyed.

To advance to the next level, you must score 50 points. If the present level is too easy, you can enter the next level by solving as few as two problems, receiving 25 points for each correct answer. If you succeed in destroying the UFO only on its third pass each time, then you will have to solve ten problems before moving to the next level. Thus, you get more practice on problems that stretch your abilities.

Bonus Points

You can earn the 50 points necessary to advance to higher levels with any combination of 5, 10, or 25 points, but you can earn bonus points for speed and accuracy. If the average score for the problems solved in a level is 25, the player receives 50 bonus points. The only way to get 50 bonus points is to score correct hits on the first two problems in a level during the first pass of the UFO. If you average 10 points or better per problem (but less than 25), you will earn 25 bonus points. There are no bonus points if you average less than 10 points per problem. Like regular points, bonus points dash onto the screen from the right.

Missile Math

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
C9 10 HOME : VTAB 10: HTAB 14: PRINT "LOADING DATA"
4C 20 GOSUB 930
4C 30 HOME
5F 40 VTAB 4:A$ = "MISSILE MATH": GOSUB 880
9C 50 PRINT
95 60 VTAB 6: PRINT "              USE PADDLE 0"
BD 70 VTAB 8
3A 80 PRINT : PRINT "         PRESS SPACE TO SELECT"
18 90 PRINT : PRINT "         PRESS RETURN TO START"
E7 100 VTAB 15: PRINT "      ADDITION &        MULTIPL
    ICATION"
EB 110 PRINT "      SUBTRACTION       & DIVISION"
0A 120 G = 0
5A 130 VTAB 18
D9 140 PRINT : PRINT "SLOW
               "
46 150 PRINT : PRINT "FAST
               "
94 160 IF G = 0 THEN VTAB 19: HTAB 9
06 170 IF G = 1 THEN VTAB 21: HTAB 9
CF 180 IF G = 2 THEN VTAB 19: HTAB 29
43 190 IF G = 3 THEN VTAB 21: HTAB 29
64 200 PRINT "*"
0F 210 POKE - 16368,0
0F 220 IF PEEK ( - 16384) < 128 THEN 220
F0 230 GET A$: IF A$ = CHR$ (13) THEN 270
F3 240 IF A$ < > " " THEN 210
E6 250 G = G + 1: IF G = 4 THEN G = 0
9A 260 GOTO 130
AF 270 L = 1:SC = 0:S = 3
16 280 IF G = 0 OR G = 2 THEN POKE 768,140: GOTO 300
```

```
DE 290 POKE 768,80
63 300 TEXT : HOME : VTAB 7: PRINT "     POSITION GUN
       OVER CORRECT ANSWER"
47 310 BN = 0:BO = 0:B = 0:SH = 0
A3 320 PRINT : PRINT : PRINT "                AND SHOOT
       UFO"
EA 330 PRINT : PRINT : PRINT "      BE CAREFUL NOT TO
       WASTE SHOTS"
59 340 PRINT : PRINT : PRINT "              ENTERING L
       EVEL ";L
18 350 FOR I = 1 TO 5000: NEXT
61 360 GR : HOME : VTAB 23: HTAB 25: PRINT "SHIPS: "
       ;S
84 370 HTAB 25: PRINT "SCORE: ";SC
06 380 A = INT ( RND (1) * 4 + 1) + 4 * (L - 1)
4E 390 B = INT ( RND (1) * 4 * L + 1)
97 400 IF G < 2 THEN C = A + B: GOTO 420
19 410 C = A * B
BA 420 A$ = STR$ (A):B$ = STR$ (B):C$ = STR$ (C)
D1 430 RP = INT ( RND (1) * 4) + 1
DF 440 IF RP = 1 THEN ANS = A:A$ = "?"
EC 450 IF RP = 2 THEN ANS = B:B$ = "?"
79 460 IF RP > 2 THEN ANS = C:C$ = "?"
49 470 VTAB 23: IF G < 2 THEN PRINT A$"+"B$"="C$: GO
       TO 490
84 480 PRINT A$"X"B$"="C$
95 490 IF ANS > 90 THEN DL = 10: GOTO 520
B9 500 IF ANS > 20 THEN DL = 5: GOTO 520
3E 510 DL = INT ((ANS / 10) + 1)
A2 520 DT = INT ( RND (1) * 5) + 1
D7 530 VTAB 21
3F 540 FOR I = 1 TO 5:CH(I) = ANS + (I - DT) * DL: H
       TAB I * 4 + 4: PRINT CH(I);: NEXT
95 550 FOR I = 1 TO 5: IF CH(I) = ANS THEN POKE 800,
       I - 1
89 560 NEXT
FC 570 CALL 24576
35 580 SH = SH + 1
CF 590 X = PEEK (769): IF X = 35 THEN 680
B9 600 IF X = 25 THEN B = 5
E9 610 IF X = 15 THEN B = 10
5E 620 IF X = 5 THEN B = 25
4F 630 HOME
E6 640 BN = BN + B
1B 650 SC = SC + B: VTAB 23:A$ = STR$ (B) + " POINTS
       ": GOSUB 880
31 660 IF BN > = 50 THEN L = L + 1: GOTO 750
A3 670 GOTO 350
24 680 S = S - 1
```

```
B2 690 HTAB 1: VTAB 21: PRINT "
                              ";:A$ = "KEEP TRYING": GOSUB
       880
F5 700 PRINT :A$ = "I KNOW YOU CAN DO IT": GOSUB 880
76 710 PRINT :A$ = "THE ANSWER WAS:": GOSUB 880: PRI
       NT :A$ = STR$ (ANS): GOSUB 880
F5 720 FOR I = 1 TO 1000: NEXT
E0 730 IF S = 0 THEN 820
9E 740 GOTO 350
70 750 BO = INT (BN / SH + .5)
9F 760 IF BO > = 25 THEN BO = 50: GOTO 790
A7 770 IF BO > = 10 THEN BO = 25: GOTO 790
25 780 GOTO 810
E5 790 VTAB 21
CF 800 A$ = "BONUS: " + STR$ (BO): GOSUB 880:SC = SC
       + BO
97 810 FOR I = 1 TO 5000: NEXT : GOTO 300
39 820 TEXT : HOME : VTAB 5: PRINT "                    S
       CORE: "SC
FB 830 PRINT : PRINT : HTAB 18: PRINT "GAME": PRINT
       : PRINT : HTAB 18: PRINT "OVER"
2F 840 VTAB 20: HTAB 5: PRINT "PRESS ANY KEY TO PLAY
       AGAIN"
1D 850 POKE - 16368,0
39 860 IF PEEK ( - 16384) < 128 THEN 860
B2 870 GET A$: GOTO 30
F2 880 X = LEN (A$):A$ = A$ + " "
2C 890 FOR I = 1 TO X + 19 - INT (X / 2 + .5)
8A 900 HTAB 39 - I: PRINT MID$ (A$,1,I);
E7 910 NEXT I
1D 920 RETURN
A2 930 CK = 0
1F 940 FOR I = 24576 TO 25329: READ A:CK = CK + A: P
       OKE I,A: NEXT
C0 950 IF CK < > 73926 THEN PRINT "ERROR IN DATA": E
       ND
25 960 RETURN
4C 970 DATA 76,22,96,255,160,160,232,169
C6 980 DATA 201,197,160,160,176,165,162,160
0F 990 DATA 128,6,10,14,18,22,32,46
25 1000 DATA 96,32,86,96,32,216,97,32
F7 1010 DATA 111,98,32,175,98,173.16,96
E5 1020 DATA 208,239,32,212,98,96,169,1
82 1030 DATA 141,16,96,141,4,96,169,2
67 1040 DATA 141,5,96,169,33,141,7,96
5D 1050 DATA 141,8,96,169,5,141,12,96
3B 1060 DATA 141,1,3,169,0,141,13,96
D1 1070 DATA 169,9,141,14,96,96,173,4
75 1080 DATA 96,205,5,96,240,34,169,0
7F 1090 DATA 32,100,248,173,4,96,141,3
```

```
4C 1100 DATA 96,32,43,97,169,6,32,100
44 1110 DATA 248,173,5,96,141,3,96,32
D5 1120 DATA 43,97,173,5,96,141,4,96
66 1130 DATA 169,0,32,100,248,173,7,96
9B 1140 DATA 141,6,96,173,12,96,141,9
F9 1150 DATA 96,32,126,97,173,14,96,32
A2 1160 DATA 100,248,173,8,96,141,6,96
03 1170 DATA 173,1,3,141,9,96,32,126
46 1180 DATA 97,173,8,96,141,7,96,173
D1 1190 DATA 1,3,141,12,96,173,13,96
C2 1200 DATA 201,1,208,68,169,0,32,100
57 1210 DATA 248,172,2,3,185,17,96,24
7B 1220 DATA 105,2,168,173,10,96,32,0
5F 1230 DATA 248,172,2,3,185,17,96,24
9D 1240 DATA 105,2,168,173,11,96,32,113
91 1250 DATA 248,201,0,208,28,169,15,32
19 1260 DATA 100,248,172,2,3,185,17,96
98 1270 DATA 24,105,2,168,173,11,96,32
9C 1280 DATA 0,248,173,11,96,141,10,96
EF 1290 DATA 96,173,32,3,205,2,3,240
20 1300 DATA 6,169,15,141,14,96,96,169
9D 1310 DATA 0,141,16,96,169,15,141,14
EA 1320 DATA 96,173,14,96,32,100,248,32
A1 1330 DATA 126,97,32,187,98,206,14,96
C5 1340 DATA 208,239,96,174,3,96,189,17
77 1350 DATA 96,24,105,2,168,169,36,133
20 1360 DATA 45,169,34,32,40,248,174,3
E7 1370 DATA 96,189,17,96,168,200,24,105
36 1380 DATA 3,133,44,169,37,32,25,248
3D 1390 DATA 174,3,96,189,17,96,168,24
AD 1400 DATA 105,4,133,44,169,38,32,25
A4 1410 DATA 248,174,3,96,189,17,96,168
61 1420 DATA 200,169,39,32,0,248,174,3
4B 1430 DATA 96,189,17,96,168,200,200,200
4B 1440 DATA 169,39,32,0,248,96,173,6
E1 1450 DATA 96,168,24,105,5,133,44,173
F4 1460 DATA 9,96,200,32,25,248,173,6
2A 1470 DATA 96,168,24,105,6,133,44,173
F6 1480 DATA 9,96,24,105,1,32,25,248
E1 1490 DATA 173,6,96,168,24,105,6,133
D0 1500 DATA 44,173,9,96,24,105,2,32
A7 1510 DATA 25,248,173,6,96,168,24,105
F3 1520 DATA 6,133,44,173,9,96,24,105
D3 1530 DATA 3,32,25,248,173,6,96,168
E0 1540 DATA 24,105,5,133,44,200,173,9
F0 1550 DATA 96,24,105,4,32,25,248,96
97 1560 DATA 162,0,32,30,251,152,201,50
8E 1570 DATA 144,17,201,100,144,18,201,150
80 1580 DATA 144,19,201,200,144,20,169,4
```

```
BD 1590 DATA 76,7,98,169,0,76,7,98
82 1600 DATA 169,1,76,7,98,169,2,76
06 1610 DATA 7,98,169,3,76,7,98,141
5E 1620 DATA 5,96,173,8,96,240,6,206
93 1630 DATA 8,96,76,55,98,169,33,141
0E 1640 DATA 8,96,32,84,98,169,9,141
C0 1650 DATA 14,96,169,0,141,13,96,173
55 1660 DATA 1,3,24,105,10,141,1,3
42 1670 DATA 201,35,208,3,76,147,98,173
CC 1680 DATA 13,96,201,1,208,48,173,11
0D 1690 DATA 96,201,2,144,15,206,11,96
13 1700 DATA 206,11,96,206,11,96,206,11
B3 1710 DATA 96,76,110,98,169,2,141,13
BD 1720 DATA 96,169,0,32,100,248,172,2
78 1730 DATA 3,185,17,96,24,105,2,168
20 1740 DATA 173,10,96,32,0,248,96,173
A7 1750 DATA 97,192,48,1,96,173,13,96
CF 1760 DATA 201,0,208,22,169,33,141,10
C9 1770 DATA 96,141,11,96,169,1,141,13
A9 1780 DATA 96,173,4,96,141,2,3,32
88 1790 DATA 198,98,96,169,0,141,16,96
82 1800 DATA 169,15,141,14,96,173,14,96
69 1810 DATA 32,100,248,32,43,97,32,187
64 1820 DATA 98,206,14,96,208,239,96,174
D7 1830 DATA 0,3,160,0,200,208,253,202
84 1840 DATA 208,250,96,162,10,160,0,200
1A 1850 DATA 208,253,202,208,250,96,162,15
05 1860 DATA 173,48,192,169,4,32,168,252
AC 1870 DATA 202,208,245,96,169,16,141,15
22 1880 DATA 96,160,1,162,1,169,80,32
EB 1890 DATA 168,252,173,48,192,232,208,253
0A 1900 DATA 136,208,240,206,15,96,208,233
38 1910 DATA 96,255,0,0,255,255,0,0
```

Reflection

Sean Puckett
Translation by Chris Poer

"Reflection" is a fast-paced computer version of reversi. You can play it as a strategy game with two people, challenge the computer, even set up your own game board. For all Apple II–series computers in DOS 3.3 or ProDOS.

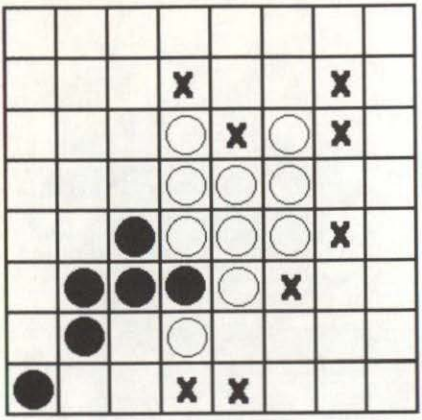
Through the ages, people have devised many pastimes to exercise their minds. The best-known match of wits is chess, with backgammon and checkers running close behind. Another board game, reversi, though perhaps not as popular now as when it was introduced in the late 1800s, does manage to combine the logic of chess, checkers, and backgammon with the action and excitement of a good football game.

The trouble is, some players can become so excited that they tend to get carried away and attempt a forward pass with the board, or they fumble and scatter the chips everywhere (a method often employed by sore losers). A computer version of reversi is ideal. The computer can act as referee, permitting only legal moves, or it can be a ruthless opponent.

"Reflection" gives you the option of playing either way—against another person or against the computer. The rules are quite simple. Players take turns placing chips on the board, one piece per turn. To capture your opponent's pieces, you sandwich a row of them between one of your existing pieces and the one you're laying down. You can capture one or several pieces this way. The row can be vertical, horizontal, or diagonal. Once a piece is captured, it turns to your color and becomes (at least temporarily) one of your pieces.

In this example, the black player can capture pieces by placing a chip on any spot marked with an X:

Multiple Choices



The best move is either the one that captures the most pieces, or the one which leaves your own pieces less vulnerable—depending on the stage of the game. Sometimes you can place a single piece to capture more than one row of chips. Each player must capture at least one enemy piece per turn, or the turn is forfeited. Corner and edge positions are valuable, since these are difficult if not impossible positions to surround. When all of one player's pieces have been captured, or when neither player can make a legal move, the chips are tallied and victory is awarded.

Because capturing an enemy piece converts it to your color, the game can reverse directions very quickly. Even if you think you're winning, you can suddenly find yourself far behind, with most of your chips flipped to your opponent's color. Reversi's sudden reversals keep players on their toes. You must plan several moves ahead, since a seemingly triumphant capture can just set you up for devastation.

Playing Reflection

Before you begin play, you have several options to choose from:

W(hite) moves first
B(lack) moves first

N(ormal) game board
D(ifferent) game board

O(ne) player
T(wo) players

What level (**1–2**)

Computer plays **B**(lack)
Computer plays **W**(hite)

Press the appropriate keys as the options appear. You have your choice of which player goes first, what board you'll use, whether you're playing against the computer (one player) or against another person (two players), the level of play (only if you're playing against the computer), and what color the computer takes (again, only against the computer).

If you're playing on the normal game board, the grid then appears, with two black and two white chips in a diamond-shaped pattern in the middle of the screen. A cursor, which repeatedly enlarges and contracts, is also visible.

If you're playing Reflection on an Apple IIe or IIc, make sure the Caps Lock key is pressed down. Use the I, K, M, and J keys to move the cursor up, right, down, and left, respectively. Press the space bar to put down your piece. You can put down only one piece per move, and only on empty squares. If you place your chip so that it doesn't capture any enemy pieces, the program removes the piece and you forfeit your turn. You must purposely forfeit in this way if you can't make a legal move. If neither player can make a move, press E to end the game.

Reversi Options

When playing against the computer, there are two levels of computer intelligence. Level 2 plays better, but it takes longer for the computer to make up its mind.

You can also set up the board yourself, placing as many pieces as you want. This is a good way to restart an unfinished game (assuming you noted the pieces' positions) or to

study a particular reversi problem. After choosing D from the option menu, an empty grid will appear. Press the W key to set down a white chip, B for a black chip, and space to skip a square. You continue left to right, top to bottom, until you reach the lower-right corner.

To end the game, simply press the E key.

Two Programs

Before you load Reflection, you need to enter two POKE statements in direct mode:

POKE 104, 64: POKE 16384,0

To make it easier, you can use Program 1, "Reflection Loader." It does the POKEs for you, then loads Reflection (Program 2) from disk. Just make sure that Reflection is on the same disk as the Loader, and that Reflection is named exactly that.

Program 1. Reflection Loader

```
10 POKE 104,64: POKE 16384,0
20 PRINT CHR$ (4)"RUN REFLECTION"
```

Program 2. Reflection

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in the following program.

```
F2 5 CLEAR : DIM BO(80),TA(71),A(71),PO(80),PT(71)
E8 10 TU = 1: ROT= 0: POKE 232,28: POKE 233,3: TEXT
     : HOME :FL = 1:PL = 1
8B 12 RESTORE
15 15 GOSUB 9000
08 20 GOSUB 1000
89 30 GOSUB 2000
8F 40 GOSUB 10000
86 50 IF DE = 1 THEN GOSUB 3000: GOTO 100
25 60 HCOLOR= 4: SCALE= 1: DRAW 2 AT 154,82: DRAW 2
     AT 128,62
01 70 HCOLOR= 7: DRAW 2 AT 127,82: DRAW 2 AT 153,62
FE 80 BC = 2:WC = 2: FOR Y = 2 TO 5: FOR X = 2 TO 5
FD 90 READ A:PO(Y * 9 + X) = A: NEXT : NEXT
08 100 FL = 1: IF TU = 1 THEN M$ = "BLACK'S TURN": G
     OTO 120
CA 110 M$ = "WHITE'S TURN"
28 120 VTAB 21: PRINT TAB( 15)M$: VTAB (23): PRINT T
     AB( 10);BC; TAB( 30);WC;"        "
```

```
32 125 VTAB (22): PRINT "   BLACK'S CHIPS      WHITE'
       S CHIPS"
D1 127 IF PL = 1 THEN AL = BC + 1: GOTO 130
73 128 AL = WC + 1
0A 130 IF COM = 1 AND TU = PL THEN GOSUB 6000: GOTO
       250
D1 135 X = 4:Y = 4:Z = 1: POKE - 16368,0
FB 140 Q = PEEK ( - 16384):X1 = X:Y1 = Y: POKE - 163
       68,0
07 150 IF Q = 201 AND Y > 0 THEN Y = Y - 1
13 160 IF Q = 205 AND Y < 7 THEN Y = Y + 1
A0 170 IF Q = 203 AND X < 7 THEN X = X + 1
9E 180 IF Q = 202 AND X > 0 THEN X = X - 1
18 183 IF Q < > 197 THEN 186
79 184 VTAB (23): PRINT "    ARE YOU SURE YOU WANT T
       O QUIT? ";: GET A$: IF A$ = "Y" THEN 7000
B5 185 VTAB (22): PRINT "
                  "
4D 186 IF BO(X1 + 9 * Y1) = 2 THEN OF = 1: GOTO 190
D0 187 OF = 0
DA 190 SCALE= Z: HCOLOR= 6: DRAW 1 AT X1 * 26 + 39,Y
       1 * 20 + 2
5B 200 IF BO(X1 + 9 * Y1) < > 0 THEN SCALE= 1: HCOLO
       R= 4 + (BO(X1 + 9 * Y1) - 2) * 3: DRAW 2 AT X
       1 * 26 + 49 + OF + (X1 > 4) * 2,Y1 * 20 + 2
9E 210 Z = Z + 3: IF Z > 16 THEN Z = 1
92 215 IF X < > X1 OR Y < > Y1 THEN POKE 768,1: POKE
        769,160: CALL 770
2A 220 SCALE= Z: HCOLOR= 5: DRAW 1 AT X * 26 + 39,Y
       * 20 + 2
D0 230 IF Q < > 160 THEN 140
5D 233 IF BO(X + 9 * Y) > 0 THEN 140
02 235 IF TU = 1 THEN OF = 1: GOTO 240
C7 237 OF = 0
32 240 SCALE= Z: HCOLOR= 6: DRAW 1 AT X * 26 + 39,Y
       * 20 + 2
EA 250 SCALE= 1: HCOLOR= 4 + (TU - 1) * 3
08 253 IF FL = 0 THEN 280
AC 255 POKE 768,2: POKE 769,110: CALL 770
3A 260 DRAW 2 AT X * 26 + 49 + OF + (X > 4) * 2,Y *
       20 + 2
79 265 POKE 768,3: POKE 769,125: CALL 770
64 267 XY = Y * 9 + X: IF PO(XY) = 0 THEN 290
CF 270 GOSUB 4000
23 280 IF CHIPS > 0 THEN GOSUB 5000:BO(XY) = TU + 1:
        GOTO 320
77 290 VTAB (23): PRINT "    FALSE MOVE, FORFEITURE
       OF TURN."
E9 295 POKE 768,50: POKE 769,10: CALL 770
78 296 FOR I = 1 TO 500: NEXT I
```

```
3F 297 IF FL = 0 THEN 340
8F 299 POKE 768,3: POKE 769,125: CALL 770
1D 300 HCOLOR= 6: DRAW 2 AT X * 26 + 49 + OF + (X >
       4) * 2,Y * 20 + 2
BA 310 POKE 768,2: POKE 769,110: CALL 770: GOTO 340
8F 320 IF TU = 1 THEN BC = BC + CHIPS + 1:WC = WC -
       CHIPS: GOTO 333
FE 330 BC = BC - CHIPS:WC = WC + CHIPS + 1
2B 333 FOR Q = 1 TO 8
BE 337 IF XY + OF(Q) > - 1 THEN PO(XY + OF(Q)) = 1
0A 338 NEXT Q
CD 340 TU = (TU - 2) * - 1 + 1
3E 350 IF WC = 0 OR BC = 0 THEN 7000
E3 360 IF WC + BC = 64 THEN 7000
4C 370 GOSUB 500
C3 380 IF XY = 0 OR XY = 7 OR XY = 63 OR XY = 70 THE
       N GOSUB 6800
8D 400 GOTO 100
0B 500 FOR I = 0 TO 71:TA(I) = 0: NEXT : RETURN
88 1000 HOME : VTAB (2): HTAB (14): INVERSE : PRINT
       "REFLECTION": NORMAL
93 1001 VTAB (4): PRINT TAB( 9)"(I-J-K-M) MOVES CURS
       OR."
E0 1002 PRINT TAB( 8)"PRESS SPACE TO MAKE MOVE."
DA 1003 PRINT : PRINT TAB( 11)"TYPE (E) TO QUIT."
66 1010 VTAB (10): PRINT TAB( 11)"(W)HITE MOVE FIRST
       "
BF 1020 PRINT TAB( 11)"(B)LACK MOVE FIRST"
D3 1030 POKE - 16368,0
06 1040 IF PEEK ( - 16384) < 128 THEN 1030
44 1050 GET A$: IF A$ = "W" THEN TU = 2: GOTO 1070
FD 1060 IF A$ < > "B" THEN 1030
37 1070 VTAB (13): PRINT TAB( 10)"(N)ORMAL GAME BOAR
       D"
A9 1080 PRINT TAB( 9)"(D)IFFERENT GAME BOARD"
EB 1090 POKE - 16368,0
58 1100 IF PEEK ( - 16384) < 128 THEN 1090
10 1110 GET A$: IF A$ = "D" THEN DE = 1: GOTO 1140
80 1120 IF A$ < > "N" THEN 1090
B9 1130 BO(30) = 2:BO(40) = 2:BO(31) = 3:BO(39) = 3
82 1140 VTAB (16): PRINT TAB( 14)"(O)NE PLAYER"
6C 1150 PRINT TAB( 14)"(T)WO PLAYERS"
E1 1160 POKE - 16368,0
5C 1170 IF PEEK ( - 16384) < 128 THEN 1170
75 1180 GET A$: IF A$ = "T" THEN RETURN
90 1190 IF A$ < > "O" THEN 1160
12 1200 COM = 1: VTAB (19): PRINT TAB( 13)"WHAT LEVE
       L (1-2)"
CF 1210 POKE - 16368,0
F1 1220 IF PEEK ( - 16384) < 128 THEN 1210
```

```
ED 1230 GET A$:LE = VAL (A$): IF LE < 1 OR LE > 3 TH
      EN 1230
B3 1240 VTAB (21): PRINT TAB( 9)"COMPUTER PLAYS (B)L
      ACK"
27 1250 PRINT TAB( 9)"COMPUTER PLAYS (W)HITE"
E3 1260 POKE - 16368,0
56 1270 IF PEEK ( - 16384) < 128 THEN 1260
C3 1280 GET A$: IF A$ = "W" THEN PL = 2: GOTO 1300
3E 1290 IF A$ < > "B" THEN 1280
C7 1300 HOME : RETURN
96 2000 HGR
91 2010 FOR I = 0 TO 159
55 2020 HCOLOR= 6: HPLOT 36,I TO 244,I
18 2023 HCOLOR= 2: HPLOT 0,I TO 33,I
0B 2026 HCOLOR= 5: HPLOT 245,I TO 279,I
78 2030 NEXT I
DD 2040 HCOLOR= 4
EE 2050 FOR I = 1 TO 8
75 2060 HPLOT I * 26 + 36,0 TO I * 26 + 36,159
4F 2070 HPLOT 36,I * 20 TO 244,I * 20
8C 2080 NEXT I
E0 2130 RETURN
B2 3000 VTAB (22): PRINT "TYPE (W) FOR PLACING A WHI
      TE CHIP HERE."
6A 3003 PRINT "TYPE (B) FOR PLACING A BLACK CHIP HER
      E."
83 3005 PRINT " HIT THE SPACEBAR TO MOVE THE CURSOR.
      "
31 3009 FOR I = 0 TO 7: FOR T = 0 TO 7
56 3010 X = T * 26 + 38:Y = I * 20 + 2
D1 3020 POKE - 16368,0
06 3030 Q = PEEK ( - 16384)
E6 3040 IF Q = 160 OR Q = 194 OR Q = 215 THEN 3080
55 3050 HCOLOR= 6: DRAW 1 AT X,Y:Z = Z + 2: IF Z > 1
      6 THEN Z = 1
5A 3060 SCALE= Z: HCOLOR= 5: DRAW 1 AT X,Y
76 3070 GOTO 3030
5B 3080 HCOLOR= 6: DRAW 1 AT X,Y
D5 3090 IF Q = 215 THEN HCOLOR= 7:OF = 0:WC = WC + 1
      :BO(T + 9 * I) = 3: GOTO 3110
CD 3100 IF Q = 194 THEN HCOLOR= 4:OF = 1:BC = BC + 1
      :BO(T + 9 * I) = 2: GOTO 3110
9D 3105 POKE 768,1: POKE 769,160: CALL 770: GOTO 312
      0
5E 3110 SCALE= 1: DRAW 2 AT X + 11 + OF + (T > 4) *
      2,Y
90 3115 POKE 768,3: POKE 769,125: CALL 770
91 3116 IF Q = 160 THEN 3120
18 3117 FOR E = 1 TO 8
```

```
DD 3110 IF T + 9 * I + OF(E) > 0 THEN PO(T + 9 * I +
        OF(E)) = 1
F1 3119 NEXT
FB 3120 NEXT T: NEXT I
D1 3130 HOME : RETURN
D9 4000 CHIPS = 0: FOR I = 1 TO 8:L = 1:V = 0
1E 4005 V = V + OF(I): IF XY + V > 70 OR XY + V < 0
       THEN 4040
83 4006 IF BO(XY + V) = 5 THEN 4040
E2 4010 IF BO(XY + V) = 4 - TU THEN XX = 1:L = L + 1
      : GOTO 4005
B4 4020 IF XX = 1 AND BO(XY + V) = TU + 1 THEN GOSUB
        4100
9B 4040 XX = 0: NEXT I
EC 4060 RETURN
74 4100 W = 1:V = 0
FA 4110 V = V + OF(I):TA(XY + V) = TU + 1
53 4120 W = W + 1: IF W < L THEN 4110
09 4130 CHIPS = CHIPS + W - 1: RETURN
EA 5000 FOR I = 0 TO 7: FOR T = 0 TO 7
F1 5010 IF TA(T + I * 9) = 0 THEN 5080
18 5020 HCOLOR= 6: DRAW 2 AT T * 26 + 49 + (T > 4) *
        2,I * 20 + 2
FE 5025 POKE 768,2: POKE 769,110: CALL 770
8E 5030 HCOLOR= 4 + (TU - 1) * 3: DRAW 2 AT T * 26 +
        49 + OF + (T > 4) * 2,I * 20 + 2
63 5040 BO(T + I * 9) = TU + 1
A0 5055 POKE 768,3: POKE 769,125: CALL 770
F9 5060 FOR Q = 1 TO 8
77 5070 IF XY + OF(Q) > 0 THEN PO(XY + OF(Q)) = 1
BB 5075 NEXT Q
14 5080 NEXT T: NEXT I
F9 5090 RETURN
44 6000 HY = - 32000:OF = (PL - 2) * - 1
57 6010 HI = - 32000: FOR XY = 0 TO 70: IF PO(XY) =
      0 OR BO(XY) > 0 THEN NEXT XY: GOTO 6203
42 6030 GOSUB 4000
26 6040 IF CHIPS = 0 THEN NEXT XY: GOTO 6203
24 6060 TT = WC + BC:QW = (TT / 8) * CHIPS + PT(XY)
     * (65 - TT) / 8
17 6065 IF LE = 2 AND CHIPS = A1 THEN QW = 10000
F2 6070 IF LE = 2 AND REC = 0 THEN GOSUB 6400: NEXT
      XY: GOTO 6203
26 6080 IF QW > HI THEN HI = QW:H1 = XY: NEXT : GOTO
        6203
F4 6100 IF HI = 0 THEN NEXT XY: GOTO 6203
AC 6110 IF QW / HI > .85 AND QW / HI < 1.15 THEN ZZ
      = INT ( RND (1) * 2): IF ZZ = 1 THEN HI = QW
      :H1 = XY
```

```
AA 6200 NEXT
6D 6203 IF LE = 2 AND REC = 1 THEN RETURN
A6 6205 IF (HI = - 32000 AND LE = 1) OR (HY = - 3200
       0 AND LE = 2) THEN FL = 0:CHIPS = 0
B6 6210 XY = H1
28 6220 IF LE = 2 THEN XY = H2
3D 6230 GOSUB 500
1E 6250 Y = INT (XY / 9):X = XY - Y * 9
F2 6260 RETURN
6D 6400 A1 = AL: FOR E = 0 TO 70
3D 6410 A(E) = BO(E)
D3 6420 IF TA(E) > 0 THEN BO(E) = TA(E):A1 = A1 + 1
80 6430 NEXT E
3F 6440 BO(XY) = TU + 1
03 6441 FOR Q = 1 TO 8
21 6442 IF XY + OF(Q) > - 1 THEN PO(XY + OF(Q)) = PO
       (XY + OF(Q)) + 1
C0 6446 NEXT Q
D9 6450 NW = QW:REC = 1:Y1 = XY
68 6460 TU = 3 - TU: GOSUB 6010:REC = 0:TU = 3 - TU:
       QW = NW - HI
F2 6470 IF QW > HY THEN HY = QW:H2 = Y1: GOTO 6550
AF 6490 IF HY = 0 THEN 6550
05 6500 IF QW / HY > .85 AND QW / HY < 1.15 THEN ZZ
       = INT ( RND (1) * 2): IF ZZ = 1 THEN HY = QW
       :H2 = Y1
55 6550 XY = Y1
03 6560 FOR E = 0 TO 70
BC 6570 BO(E) = A(E)
D0 6580 NEXT
5B 6590 GOSUB 500
EE 6600 FOR Q = 1 TO 8
36 6610 IF Y1 + OF(Q) < 0 THEN 6630
87 6615 IF PO(Y1 + OF(Q)) = 2 THEN PO(Y1 + OF(Q)) =
       1: GOTO 6630
34 6620 PO(Y1 + OF(Q)) = 0
90 6630 NEXT Q
F2 6640 RETURN
28 6800 IF XY = 7 THEN 6860
F0 6810 IF XY = 63 THEN 6890
DD 6820 IF XY = 70 THEN 6920
E9 6830 FOR I = 9 TO 13:PT(I) = 15 - I: NEXT
E2 6840 FOR I = 1 TO 37 STEP 9:PT(I) = 6 - INT (I /
       9): NEXT
FA 6850 RETURN
64 6860 FOR I = 6 TO 42 STEP 9:PT(I) = 6 - INT (I /
       9): NEXT
57 6870 FOR I = 16 TO 12 STEP - 1:PT(1) = I - 19
07 6880 RETURN
F1 6890 FOR I = 54 TO 59:PT(I) = I - 48: NEXT
```

```
45 6900 FOR I = 64 TO 28 STEP - 9:PT(I) = INT (I / 9
     ) - 1: NEXT
EC 6910 RETURN
F4 6920 FOR I = 62 TO 58 STEP - 1:PT(I) = I - 57: NE
     XT
44 6930 FOR I = 69 TO 33 STEP - 9:PT(I) = INT (I / 9
     ) - 1: NEXT
F8 6940 RETURN
3F 7000 SCALE= 1:WI = 3: IF WC > BC THEN GC = WC:BL
     = 4:M$ = "WHITE":WH = 1: GOTO 7020
F3 7010 IF BC > WC THEN GC = BC:WI = 2:WH = 3:M$ = "
     BLACK":BL = 6: GOTO 7020
30 7015 TI = 1:GC = WC:WH = 1:BL = 6:WI = 0
51 7020 FOR I = 1 TO GC
BC 7030 IF WC > = I THEN HCOLOR= 3: DRAW 2 AT 15,140
     - I * 2
AB 7040 IF BC > = I THEN HCOLOR= 4: DRAW 2 AT 266,14
     0 - I * 2
41 7045 POKE 768,2: POKE 769,80 + I * 2: CALL 770
85 7050 NEXT I
57 7060 HCOLOR= CO: FOR I = 1 TO GC
EF 7070 IF WC > = I THEN HCOLOR= WH: DRAW 2 AT 15,14
     0 - I * 2
6F 7075 IF BC > = I THEN HCOLOR= BL: DRAW 2 AT 266,1
     40 - I * 2
29 7080 POKE 768,2: POKE 769,80 + I * 2: CALL 770
95 7090 NEXT I
3D 7100 HOME : VTAB (21): IF TI THEN PRINT TAB( 10)"
     THE GAME IS A TIE": GOTO 7120
DF 7110 PRINT TAB( 12)M$" IS THE WINNER"
FF 7120 PRINT "  WOULD YOU LIKE TO PLAY AGAIN? (Y/N)
     ";
DB 7130 POKE - 16368,0
2E 7140 IF PEEK ( - 16384) < 128 THEN 7130
57 7150 GET A$: IF A$ = "N" THEN TEXT : HOME : END
DE 7160 IF A$ < > "Y" THEN 7150
C8 7170 GOTO 5
CC 7200 END
E1 9000 FOR I = 1 TO 8
E1 9010 READ A
7C 9020 OF(I) = A
7F 9030 NEXT I
9D 9040 FOR X = 0 TO 71
02 9050 READ A:PT(X) = A
C1 9060 NEXT
1F 9070 FOR I = 770 TO 795: READ M: POKE I,M: NEXT I
07 9080 FOR I = 8 TO 71 STEP 9:BO(I) = 5: NEXT
46 9099 RETURN
97 9100 DATA -10,-9,-8,-1,1,8,9,10
```

```
94 9120 DATA 16,-8,5,2,2,5,-8,16,0,-8,-12,-2,-2,-2,-
     2,-12,-8,0
95 9130 DATA 5,-2,8,2,2,8,-2,5,0,2,-2,2,1,1,2,-2,2,0
B7 9140 DATA 2,-2,2,1,1,2,-2,2,0,5,-2,8,2,2,8,-2,5,0
CA 9150 DATA -8,-12,-2,-2,-2,-2,-12,-8,0,16,-8,5,2,2
     ,5,-8,16,0
A4 9160 DATA  172,01,03,174,01,03,169,04,32,168,252,
     173,48,192,232,208,253,136,208,239,206,0,03,
     208,231,96
A3 10000 X = 795: IF PEEK (796) = 2 THEN RETURN
0E 10010 READ A: IF A = - 1 THEN RETURN
72 10020 X = X + 1: POKE X,A
82 10030 GOTO 10010
D6 10040 DATA 2,0,6,0,9,0
72 10050 DATA 46,60,0
86 10055 DATA 7,63,63,19,45,45,45,45
4F 10060 DATA 45,19,63,63,63,63,63,17,27,45,45,45,45
     ,45,45,45,19
0A 10070 DATA 63,63,63,63,63,63,63
20 10080 DATA 19,45,45,45,45,45,45,45,45,45,19,63,63
     ,63
D9 10090 DATA 63,63,63,63,63,63,21,45,45,45,45,45,45
     ,45,45
97 10100 DATA  45,19,63,63,63,63,63,63,63,63
13 10110 DATA 63,17,13,45,45,45,45,45,45,45,45
89 10120 DATA 19,31,63,63,63,63,63,63,63
E7 10130 DATA 10,45,45,45,45,45,45,45,19
06 10140 DATA 31,63,63,63,63,63,10,45,45,45,45,45,19
49 10150 DATA 31,63,63,63
35 10999 DATA 0,-1
F5 11000 DATA 1,1,1,1,1,0,0,1,1,0,0,1,1,1,1,1
```

Build a Quiz

Clark and Kathy H. Kidd

This versatile program lets you generate quizes and short tests on any subject. You can write true/false, multiple-choice, and completion questions, save the quiz to disk, and take it as many times as you want. For any Apple II computer using either DOS 3.3 or ProDOS.

You can buy an assortment of commercial educational programs that teach anything from Mendelian genetics to the inner workings of an automobile engine. Unfortunately, that costs money, a lot of money. And you can't expect even the best educational program to drill your children on exactly what they're studying in school. Even more important, your child's needs change from day to day, month to month. One week your child may be studying the geography of South America, the next the history of South Dakota. "Build a Quiz" can help you keep pace with those needs.

With Build a Quiz, you can either build a quiz on any subject and save it to disk, or give your child (or yourself) a test on any subject for which you've previously written and saved a quiz.

If your child isn't doing well in history, say, and there's a big test coming up, you can make up a sample test from the textbook or class notes and let him or her take the test until the material is mastered. Because this is your test, you can make it as general or specific as you wish. You can build a quiz about the U.S. Constitution or about the history of government in Texas. Whatever your child needs can be covered with Build a Quiz, and you can create tests using multiple-choice, true/false, or completion questions—or a combination of the three.

Build a Quiz has another function, too—fun. Maybe your teenager has agreed to pass your quiz on geometry if you pass a quiz about rock stars. Both tests can be constructed with this program.

Making Quizzes, Taking Quizzes

Using Build a Quiz is simple. After you've typed it in and saved it, load and run it. You'll soon be presented with three choices:

1 — BUILD A NEW QUIZ
2 — TAKE AN EXISTING QUIZ
3 — EXIT FROM THIS PROGRAM

If this is your first time with Build a Quiz, you'll choose option 1. Provide a filename for the quiz's disk data file (from 1 to 10 characters long) and hit Return. You also need to enter a quiz title, which will appear at the top of the screen when someone later takes the test. The title can be from 1 to 40 characters long.

Now you can begin typing in your questions and their answers. The menu on the screen should read:

1 — TRUE/FALSE
2 — MULTIPLE CHOICE
3 — COMPLETION
4 — QUIZ COMPLETE

Go ahead and enter as many questions as you want, mixing the types of questions as much as you want. After typing in a question, you'll be asked for its answer. True/False is obvious; Multiple Choice requires five possible answers (A–E); keep the Completion answers short so that they'll fit on one line when the quiz taker enters an answer. When you're through, hit 4 and the file is written to disk (just make sure there's one in the drive).

A special note: While Build a Quiz is a versatile program, there are some limitations to the way the computer will accept information. When building a quiz, don't use quotation marks or colons, as they'll result in an EXTRA IGNORED error.

Now that you have a quiz or two on disk, you can select option 2 from the first menu. The questions will be presented in the order in which they were entered. (Keep that in mind as you build quizzes if order is important.)

Press T or F for True/False, A–E for Multiple Choice, and type in the correct short answer for Completion. The latter's answers must be *exactly* what was entered; that's really the only limitation of Build a Quiz. When the quiz is finished, the final score is shown, both in the number right versus number of questions, and as a percentage score.

When you want to quit using Build a Quiz, choose option 3 from the main menu.

Build a Quiz

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
EB 100 REM *** BUILD A QUIZ
07 110 D$ = CHR$ (4): DIM QH$(4)
B1 120 TEXT : HOME : GR
2E 130 HTAB 9: VTAB 22
5F 140 PRINT "B U I L D    A    Q U I Z"
D9 150 GOSUB 9500
AA 160 HTAB 11: VTAB 24: INVERSE
C5 170 PRINT "PRESS ANY KEY TO START";
D4 180 NORMAL
58 190 COLOR= INT ( RND (1) * 10) + 6
C8 200 X = INT ( RND (1) * 36)
CB 210 Y = INT ( RND (1) * 23)
0B 220 VLIN Y,Y + 3 AT X: HLIN X,X + 4 AT Y: VLIN Y,
       Y + 8 AT X + 4
47 230 HLIN X + 2,X + 4 AT Y + 8: VLIN Y + 8,Y + 13
       AT X + 2: VLIN Y + 16,Y + 17 AT X + 2
CD 235 FOR X = 1 TO 300: NEXT
BF 240 X = PEEK ( - 16384): IF X < 127 THEN 190
17 250 POKE - 16368,0
61 260 TEXT : HOME
D4 270 PRINT : PRINT "OPTION 1 OF BUILD A QUIZ ALLOW
       S YOU TO"
5D 280 PRINT "CONSTRUCT YOUR OWN QUIZ AND SAVE IT AS
       "
C3 290 PRINT "A DISK DATA FILE.  THE QUIZ CAN CONTAI
       N"
04 300 PRINT "TRUE/FALSE, COMPLETION AND MULTIPLE"
F6 310 PRINT "CHOICE QUESTIONS IN ANY COMBINATION."
14 320 PRINT : PRINT : PRINT : PRINT "OPTION 2 OF BU
       ILD A QUIZ ALLOWS YOU TO"
13 330 PRINT "TEST YOURSELF WITH A QUIZ THAT ALREADY
       "
3E 340 PRINT "EXISTS AS A DISK DATA FILE."
68 350 PRINT : PRINT : PRINT : PRINT "ENTER THE OPTI
       ON YOU DESIRE:"
82 360 PRINT : PRINT "    ";: INVERSE : PRINT "1";:
       NORMAL : PRINT " - BUILD A NEW QUIZ"
4C 370 PRINT : PRINT "    ";: INVERSE : PRINT "2";:
       NORMAL : PRINT " - TAKE AN EXISTING QUIZ"
73 380 PRINT : PRINT "    ";: INVERSE : PRINT "3";:
       NORMAL : PRINT " - EXIT FROM THIS PROGRAM"
0C 390 X = PEEK ( - 16384): IF X < 128 THEN 390
0F 400 POKE - 16368,0
```

```
D1 410 X = X - (128 + 48): IF X < 1 THEN 430
FB 420 ON X GOTO 1000,3000,500
79 430 PRINT CHR$ (7);
9F 440 GOTO 390
2B 500 REM *** ALL DONE
4A 510 HOME
D7 520 GOSUB 9500
93 530 END
A1 1000 REM *** BUILD A NEW QUIZ
FE 1010 QU = 0
40 1020 HOME
1B 1030 VTAB 9: HTAB 1
39 1040 PRINT "ENTER THE NAME UNDER WHICH THIS QUIZ
        WILL BE SAVED ON THE DISK:"
9E 1050 VTAB 11: HTAB 1
02 1060 PRINT "(1 TO 10 CHARACTERS IN LENGTH)"
8B 1070 VL = 13:HL = 1
65 1080 GOSUB 11000
9F 1090 IF LEN (X$) < 1 OR LEN (X$) > 10 THEN PRINT
    CHR$ (7);: GOTO 1020
E8 1100 FL$ = "QUIZ." + X$: ONERR GOTO 15000
66 1110 PRINT : PRINT D$;"OPEN ";FL$
6E 1120 POKE 216,0: HOME
1D 1130 VTAB 9: HTAB 1
24 1140 PRINT "ENTER THE TITLE OF THE QUIZ WHICH WIL
    L  APPEAR AT THE TOP OF THE SCREEN:"
A0 1150 VTAB 11: HTAB 1
C4 1160 PRINT "(1 TO 40 CHARACTERS IN LENGTH)"
8D 1170 VL = 13:HL = 1
67 1180 GOSUB 11000
AB 1190 IF LEN (X$) < 1 OR LEN (X$) > 40 THEN PRINT
    CHR$ (7);: GOTO 1120
85 1195 GOSUB 9600:X$ = Y$
9C 1200 PRINT : PRINT D$;"WRITE ";FL$
DF 1210 PRINT CHR$ (34);X$; CHR$ (34)
99 1220 PRINT D$
E9 1300 REM *** DETERMINE QUESTION TYPE
42 1310 HOME
18 1320 VTAB 4: HTAB 1
98 1330 PRINT "ENTER QUESTION TYPE:"
01 1340 PRINT : PRINT : PRINT "     ";: INVERSE : PRI
    NT "1";: NORMAL : PRINT " - TRUE/FALSE"
9D 1350 PRINT : PRINT : PRINT "     ";: INVERSE : PRI
    NT "2";: NORMAL : PRINT " - MULTIPLE CHOICE"
52 1360 PRINT : PRINT : PRINT "     ";: INVERSE : PRI
    NT "3";: NORMAL : PRINT " - COMPLETION"
88 1370 PRINT : PRINT : PRINT "     ";: INVERSE : PRI
    NT "4";: NORMAL : PRINT " - QUIZ COMPLETE"
AB 1380 X = PEEK ( - 16384): IF X < 128 THEN 1380
F1 1390 POKE - 16368,0
```

```
A6 1400 X = X - (128 + 48)
50 1410 IF X < 1 OR X > 4 THEN PRINT CHR$ (7);: GOTO
        1380
E7 1420 QT$ = CHR$ (X + 48)
50 1430 IF X < > 4 THEN 1460
A5 1440 PRINT : PRINT D$;"WRITE ";FL$: PRINT QT$: PR
      INT D$;"CLOSE ";FL$
ED 1450 GOTO 260
58 1460 HOME
2D 1470 VTAB 3: HTAB 1
A7 1480 QU = QU + 1
41 1490 PRINT "QUESTION NUMBER ";QU
E4 1500 ON X GOTO 1600,1900,2200
33 1600 REM *** BUILD T/F QUESTION
1B 1610 VTAB 5: HTAB 1
A0 1620 PRINT "TRUE OR FALSE:"
4E 1630 VL = 7:HL = 1
61 1640 GOSUB 11000
7D 1650 QU$ = X$
34 1660 VTAB 13: HTAB 4
CA 1670 PRINT "ENTER ";: INVERSE : PRINT "T";: NORMA
      L : PRINT " FOR TRUE."
40 1680 VTAB 15: HTAB 4
40 1690 PRINT "ENTER ";: INVERSE : PRINT "F";: NORMA
      L : PRINT " FOR FALSE."
0D 1700 X = PEEK ( - 16384): IF X < 128 THEN 1700
D9 1710 POKE - 16368,0
BD 1720 X = X - 128
F1 1730 X$ = CHR$ (X)
71 1740 GOSUB 9600
46 1750 IF Y$ < > "T" AND Y$ < > "F" THEN PRINT CHR$
        (7);: GOTO 1700
BE 1760 PRINT : PRINT D$;"WRITE ";FL$
B7 1770 PRINT QT$
92 1780 PRINT CHR$ (34);QU$; CHR$ (34)
E9 1790 PRINT Y$
9D 1800 PRINT D$
64 1810 GOTO 1300
39 1900 REM *** BUILD MULTIPLE CHOICE
21 1910 VTAB 5: HTAB 1
B0 1920 PRINT "MULTIPLE CHOICE:"
54 1930 VL = 7:HL = 1
67 1940 GOSUB 11000
83 1950 QU$ = X$
16 1960 J = 0
3F 1970 FOR I = 10 TO 22 STEP 3
79 1980 VTAB I: HTAB 4: INVERSE
FD 1990 PRINT CHR$ (J + 65);".";
3A 2000 NORMAL
A4 2010 VL = I:HL = 7
```

```
4E 2020 GOSUB 11000
42 2030 QH$(J) = X$
0B 2040 J = J + 1
B6 2050 NEXT
AA 2060 VTAB 24: HTAB 1
B5 2070 PRINT "ENTER CORRECT ANSWER  (A,B,C,D OR E):
        ";
65 2080 X = PEEK ( - 16384): IF X < 128 THEN 2080
EC 2090 POKE - 16368,0
2D 2100 X$ = CHR$ (X - 128)
5A 2110 GOSUB 9600
3D 2120 IF Y$ < "A" OR Y$ > "E" THEN PRINT CHR$ (7);
        : GOTO 2080
6D 2130 HTAB 1: VTAB 1
AB 2140 PRINT : PRINT D$;"WRITE ";FL$
BC 2150 PRINT QT$: PRINT QU$
3A 2160 FOR X = 0 TO 4
DE 2170 PRINT CHR$ (34);QH$(X); CHR$ (34)
0C 2180 NEXT : PRINT Y$: PRINT D$
77 2190 GOTO 1300
47 2200 REM *** BUILD COMPLETION
14 2210 VTAB 5: HTAB 1
7C 2220 PRINT "COMPLETION:"
47 2230 VL = 7:HL = 1
5A 2240 GOSUB 11000
A1 2250 VTAB 10: HTAB 1
CC 2260 PRINT "ENTER CORRECT ANSWER:"
7E 2270 QU$ = X$
93 2280 VL = 12:HL = 1
6E 2290 GOSUB 11000
9F 2300 PRINT : PRINT D$;"WRITE ";FL$
98 2310 PRINT QT$
A0 2320 PRINT QU$
C8 2330 PRINT X$
A4 2340 PRINT D$
6B 2350 GOTO 1300
91 3000 REM *** TAKE AN EXISTING QUIZ
01 3010 QU = 0
63 3020 CO = 0
46 3030 HOME
21 3040 VTAB 9: HTAB 1
F6 3050 PRINT "ENTER THE NAME UNDER WHICH THIS QUIZ
          EXISTS ON THE DISK:"
A4 3060 VTAB 11: HTAB 1
08 3070 PRINT "(1 TO 10 CHARACTERS IN LENGTH)"
91 3080 VL = 13:HL = 1
6B 3090 GOSUB 11000
8B 3100 IF LEN (X$) < 1 OR LEN (X$) > 10 THEN PRINT
       CHR$ (7);: GOTO 3030
EE 3110 FL$ = "QUIZ." + X$: ONERR GOTO 15000
```

```
6C 3120 PRINT : PRINT D$;"OPEN ";FL$
ED 3130 POKE 216,0: PRINT D$;"READ ",FL$
58 3135 ONERR GOTO 14000
CF 3140 INPUT TI$
D5 3145 POKE 216,0
7C 3150 X$ = " " + TI$ + " ":TI$ = X$
D4 3160 IF LEN (TI$) < 40 THEN 3150
11 3170 IF LEN (TI$) > 40 THEN TI$ = LEFT$ (TI$,40)
5C 3180 HOME
2D 3190 FOR X = 130 TO 160 STEP 5
82 3200 POKE 864,4: POKE 865,X: CALL 866: NEXT
3B 3300 REM *** READ A QUESTION
16 3310 INPUT X$
88 3320 X = VAL (X$)
BD 3330 IF X < 1 OR X > 3 THEN 6000
50 3340 HOME
8A 3350 VTAB 1: HTAB 1: INVERSE
43 3360 PRINT TI$;
5D 3370 NORMAL
A7 3380 QU = QU + 1
35 3390 VTAB 3: HTAB 1
1F 3400 PRINT "QUESTION NUMBER ";QU
A6 3410 ON X GOTO 3500,3700,4000
83 3500 REM *** PRESENT TRUE FALSE
1B 3510 VTAB 5: HTAB 1
A0 3520 PRINT "TRUE OR FALSE:"
25 3530 VTAB 7: HTAB 1
26 3540 INPUT X$
B3 3550 PRINT X$;
30 3560 VTAB 11: HTAB 4
DA 3570 PRINT "ENTER ";: INVERSE : PRINT "T";: NORMA
    L
91 3580 PRINT " FOR TRUE."
40 3590 VTAB 13: HTAB 4
B2 3600 PRINT "ENTER ";: INVERSE : PRINT "F";: NORMA
    L
6E 3610 PRINT " FOR FALSE."
97 3620 X = PEEK ( - 16384): IF X < 128 THEN 3620
E1 3630 POKE - 16368,0
48 3640 X$ = CHR$ (X - 128)
75 3650 GOSUB 9600
4B 3660 IF Y$ < > "T" AND Y$ < > "F" THEN PRINT CHR$
    (7);: GOTO 3620
34 3670 INPUT X$
DA 3680 IF X$ = Y$ THEN 5000
8A 3690 GOTO 5500
68 3700 REM *** PRESENT MULTIPLE CHOICE
1F 3710 VTAB 5: HTAB 1
AE 3720 PRINT "MULTIPLE CHOICE:"
29 3730 VTAB 7: HTAB 1
```

```
2A 3740 INPUT X$
B7 3750 PRINT X$;
14 3760 J = 0
3D 3770 FOR I = 10 TO 22 STEP 3
77 3780 VTAB I: HTAB 4: INVERSE
FB 3790 PRINT CHR$ (J + 65);".";
18 3800 NORMAL : PRINT " ";
20 3810 INPUT X$
AD 3820 PRINT X$;
18 3830 J = J + 1
C3 3840 NEXT
B7 3850 VTAB 24: HTAB 1
C2 3860 PRINT "ENTER CORRECT ANSWER  (A,B,C,D OR E):
      ";
B5 3870 X = PEEK ( - 16384): IF X < 128 THEN 3870
F9 3880 POKE - 16368,0
60 3890 X$ = CHR$ (X - 128)
67 3900 GOSUB 9600
A2 3905 IF Y$ < "A" OR Y$ > "E" THEN PRINT CHR$ (7);
     : GOTO 3870
22 3910 INPUT X$
C8 3920 IF X$ = Y$ THEN 5000
78 3930 GOTO 5500
C7 4000 REM *** PRESENT COMPLETION
12 4010 VTAB 5: HTAB 1
7A 4020 PRINT "COMPLETION:"
1C 4030 VTAB 7: HTAB 1
1D 4040 INPUT X$
AA 4050 PRINT X$;
A3 4060 VTAB 10: HTAB 1
CE 4070 PRINT "ENTER CORRECT ANSWER:"
91 4080 VL = 12:HL = 1
6C 4090 GOSUB 11000
58 4100 GOSUB 9600
47 4110 QT$ = Y$
17 4120 INPUT X$
64 4130 GOSUB 9600
10 4140 IF QT$ = Y$ THEN 5000
71 4150 GOTO 5500
93 5000 REM *** CORRECT ANSWER
D8 5010 CO = CO + 1
A2 5020 FOR X = 160 TO 175 STEP 5
6C 5030 POKE 864,2: POKE 865,X: CALL 866: NEXT
D3 5040 FOR X = 1 TO 2500: NEXT
6A 5050 GOTO 3300
CB 5500 REM *** INCORRECT ANSWER
A3 5510 VTAB 24: HTAB 1
90 5520 PRINT SPC( 39);
8B 5530 VTAB 24: HTAB 1: FLASH
71 5540 GOSUB 9600
```

```
A5 5550 PRINT "*** ANSWER = "; LEFT$ (Y$,26);
1A 5560 FOR X = 90 TO 70 STEP - 5
96 5570 POKE 864,3: POKE 865,X: CALL 866: NEXT
ED 5580 FOR X = 1 TO 2500: NEXT
6B 5590 NORMAL
62 5600 GOTO 3300
9C 6000 REM *** GAME OVER
67 6010 HTAB 1: VTAB 1
25 6020 PRINT : PRINT D$;"CLOSE ";FL$
49 6030 HOME
B2 6035 FOR X = 1 TO 5: PRINT CHR$ (7);: NEXT
85 6040 VTAB 3: HTAB 1: INVERSE
5E 6050 PRINT "***    Q U I Z    C O M P L E T E
      ***";
56 6060 NORMAL
C4 6070 HTAB 5: VTAB 6
FC 6080 PRINT "   TOTAL QUESTIONS        "; RIGHT$ ("
        " + STR$ (QU),3)
4D 6090 HTAB 5: VTAB 8
93 6100 PRINT "   QUESTIONS CORRECT      "; RIGHT$ ("
        " + STR$ (CO),3)
AA 6110 IF QU = 0 THEN QU = 1
6B 6120 X = INT ((CO * 100) / QU)
CD 6130 HTAB 5: VTAB 10
B8 6140 PRINT "   PERCENT CORRECT        "; RIGHT$ ("
        " + STR$ (X),3);"%"
2A 6150 HTAB 8: VTAB 15: INVERSE
C3 6160 PRINT "PRESS ANY KEY TO CONTINUE"
5C 6170 NORMAL
EC 6180 X = PEEK ( - 16384): IF X < 128 THEN 6180
F2 6190 POKE - 16368,0
DA 6200 GOTO 260
6D 9500 REM SUBROUTINE TO PLAY SONGS
43 9510 RESTORE
F8 9520 READ QX$: IF QX$ < > "$SONG" THEN 9520
65 9530 FOR X = 866 TO 891: READ Y: POKE X,Y: NEXT
35 9540 READ X,Y: IF X < 0 THEN RETURN
A1 9550 POKE 864,Y: POKE 865,X: CALL 866: GOTO 9540
FD 9560 DATA "$SONG",172,97,3,174,97,3,232,208,253,1
      69,4,32,168,252,173,48,192,136,208,239,206,9
      6,3,208,231,96
6F 9570 DATA 134,4,151,4,151,4,151,4,166,3,177,8,166
      ,4,151,4,134,8,-1,0
12 9600 REM *** UPPER-CASE
09 9610 Y$ = ""
48 9620 FOR X = 1 TO LEN (X$)
BE 9630 Y = ASC ( MID$ (X$,X,1))
D7 9640 IF Y > 96 AND Y < 123 THEN Y = Y - 32
E5 9650 Y$ = Y$ + CHR$ (Y)
CD 9660 NEXT
```

```
02 9670 RETURN
3E 11000 REM *** GET A REPLY
7B 11010 X$ = ""
4F 11020 HTAB HL: VTAB VL
CE 11030 PRINT X$;: INVERSE : PRINT " ";: NORMAL
60 11040 X = PEEK ( - 16384): IF X < 128 THEN 11040
69 11050 POKE - 16368,0
31 11060 X = X - 128
C1 11070 IF X = 13 THEN 11300
1B 11080 IF X = 8 THEN 11200
53 11090 X$ = X$ + CHR$ (X)
84 11100 GOTO 11020
26 11200 IF LEN (X$) < 1 THEN GOTO 11040
4F 11210 HTAB HL: VTAB VL
9F 11220 PRINT SPC( LEN (X$) + 1);
60 11230 IF LEN (X$) = 1 THEN X$ = "": GOTO 11020
E4 11240 X$ = LEFT$ (X$, LEN (X$) - 1)
B0 11250 GOTO 11020
0A 11300 IF LEN (X$) > 80 THEN X$ = LEFT$ (X$,80)
A3 11305 IF LEN (X$) = 0 THEN X$ = " "
53 11310 HTAB HL: VTAB VL
CF 11320 PRINT X$;" ";
79 11330 RETURN
3C 14000 REM *** FILE OPEN ERROR
F2 14010 POKE 216,0
F7 14020 PRINT : PRINT D$;"CLOSE ";FL$
47 14030 PRINT D$;"DELETE ";FL$
F2 14040 PRINT D$
50 14050 HOME
3B 14060 FOR X = 1 TO 8: PRINT CHR$ (7);: NEXT
04 14070 VTAB 4: HTAB 1
3B 14080 PRINT "SPECIFIED FILE NOT FOUND ON DISK."
E8 14090 VTAB 6: HTAB 5: INVERSE
05 14100 PRINT "PRESS ANY KEY TO CONTINUE"
4B 14110 X = PEEK ( - 16384): IF X < 128 THEN 14110
5B 14120 POKE - 16368,0
46 14130 NORMAL
7F 14140 GOTO 260
EA 15000 REM *** INVALID FILENAME
F4 15010 POKE 216,0
3A 15020 HOME
25 15030 FOR X = 1 TO 8: PRINT CHR$ (7);: NEXT
ED 15040 VTAB 4: HTAB 1
2C 15050 PRINT "SPECIFIED FILE NAME IS INVALID."
D2 15060 VTAB 6: HTAB 5: INVERSE
3B 15070 PRINT "PRESS ANY KEY TO CONTINUE"
9C 15080 X = PEEK ( - 16384): IF X < 128 THEN 15080
91 15090 POKE - 16368,0
30 15100 NORMAL
69 15110 GOTO 260
```

4

Apple Graphics

Apple SuperFont
Custom Character Set Graphics for the Apple

Tim Victor

"Apple SuperFont" is a significant graphics enhancement for Apple II–family computers. You can now place upper- and lowercase text anywhere on the high-resolution screen. In addition, you're not limited to the built-in character set—you can easily define foreign character sets, italics, boldface, and underline fonts, as well as shapes for high-speed animated games in BASIC. The program requires a 48K or 64K Apple II+, Apple IIe, or Apple IIc, with either DOS 3.3 or ProDOS.

Without resorting to machine language, programming high-speed graphics is difficult on the Apple. High-resolution graphics look nice, but shape tables are too slow for most animation purposes. One alternative is to use character graphics for animation. Characters can move a whole block (character position) at a time and can be placed on the screen with a simple PRINT statement. Unfortunately, ordinary Apple characters aren't very suitable for games or even business charts.

Now there's a way around these problems. With "Apple SuperFont" and its accompanying utility programs, you can easily redefine a character into practically any shape you want and print it directly on the hi-res graphics screen. Custom character sets are a snap to design, and fast animation is as simple as printing a character, erasing it, and printing it again in a new location.

Several programs already exist for printing characters on the hi-res screen, including "HRCG" (High Resolution Character Generator), which is part of the Apple *DOS Tool Kit*. The Apple SuperFont "HROUT" program works much like HRCG, putting characters on the high-resolution screen from a table of character images, but the Apple SuperFont system is much more versatile.

The Apple SuperFont Editor makes it easy for you to create character sets (fonts) for use with HRCG or HROUT. Special features help you design multicharacter shapes and allow you to see the effects of the Apple's unusual use of color in hi-res graphics. Once you've created or customized a character set, you can easily use these fonts in your own programs.

Typing Apple SuperFont

To run SuperFont, you need to have four files on the same disk: APPLEFONT, APPLEFONT2, HROUT, and NORMAL.SET. There are two different versions of APPLEFONT. Program 1 is for using SuperFont with DOS 3.3. Program 2 shows the slight changes necessary to use Program 1 with ProDOS. The other three files need no modifications to work with either disk operating system.

APPLEFONT2 (Program 3, the Apple SuperFont Editor), NORMAL.SET (Program 4), and HROUT (Program 5) are all machine language binary files and must be entered with "AppleMLX," COMPUTE! Publications' machine language entry program. Don't worry, it's easy—you don't need to understand machine language to use AppleMLX or the other three programs.

To type in APPLEFONT2, NORMAL.SET, and HROUT, you first must have a working copy of AppleMLX on disk. AppleMLX insures that your typing time is well spent, for it checks for almost every possible typing error, from skipping a line to transposing characters. (To make sure you enter AppleMLX correctly, you can use another error-checking program, "Apple Automatic Proofreader," to type it in. See Appendix B for the Proofreader and its instructions.)

Read Appendix C, and type in and save AppleMLX. Make sure you save a copy of AppleMLX, for you'll use it to type in several other programs in this book, as well as Apple machine language programs found in *COMPUTE!* magazine and *COMPUTE!'s Apple Applications* issues.

When you load and run AppleMLX, it asks for starting and ending addresses for the program you're entering. Each of the three machine language programs of SuperFont have different addresses you need to provide:

APPLEFONT2 (Program 3)

```
STARTING ADDRESS? 1000
ENDING ADDRESS?   1FDF
```

NORMAL.SET (Program 4)
STARTING ADDRESS? 8D00
ENDING ADDRESS? 8FFF

HROUT (Program 5)
STARTING ADDRESS? 0300
ENDING ADDRESS? 0357

Once you finish typing in a program, save it using AppleMLX's Save command. **Important:** Make sure you name the files exactly as they are written here (APPLEFONT2, NORMAL.SET, and HROUT), or the loader program (APPLEFONT) won't recognize them.

When all the files (APPLEFONT, with ProDOS changes if appropriate; APPLEFONT2, NORMAL.SET, and HROUT) are entered and saved to disk, type RUN APPLEFONT. APPLEFONT first checks to see which operating system is in your Apple. If the correct operating system for this version of APPLEFONT is present, it will BLOAD the other three files and connect HROUT to the standard character output routine. APPLEFONT2, the SuperFont Editor, is started with a CALL to 4096. From then on, the SuperFont Editor is in complete control except when it needs to access the disk drive. If you ask to load or save a character set, control returns to the BASIC program, the file is transferred using BASIC's disk access commands, and the SuperFont Editor program is CALLed again.

Using the SuperFont Editor

Characters are designed and edited on a grid that represents 32 (vertical) × 55 (horizontal) pixels. Each cell in the grid is a fourfold enlargement of actual size. Individual cells can be turned on (white) or off (black) with the bit-editing functions, and blocks of cells can be copied from one place to another on the screen. Patterns of 7 × 8 cells can be saved from the screen to the character set being edited with the Put command. The Get command does just the reverse, pulling a character from the character set onto the editing screen.

All the features of the Editor are controlled with a series of four menus, entitled Bit Edit, Charsets, Utility, and Display. Each of these menus contains three to six selections. Only one menu is displayed on the screen at a time.

To change menus, press the space bar. The next menu title will be printed on the screen, along with the menu selections.

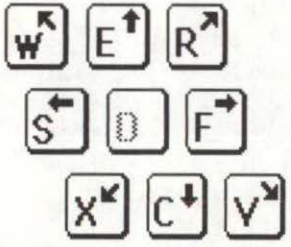
The top selection will be printed in inverse characters to indicate that it's been chosen. To select a different item, use the left- and right-arrow keys. The large cursor bar moves up or down the menu to show you which selection is active.

Some menu items, like Clear Screen or Save Set, wait for you to press the Return key before performing their functions.

Three Cursors

You'll be using three visually distinctive cursors in the SuperFont Editor: the bit cursor, the box cursor, and the character cursor. When a menu item is selected, one of the cursors may begin to flash, indicating that it can be moved. The cursors are controlled by a keypad centered on the D key:

Cursors



The bit cursor is a 1 × 1 cell box displayed on the editing screen. It flashes whenever the Bit Edit menu is displayed. Moving the bit cursor around on the editing screen sets (white) or clears (black) the cells that the cursor passes over. In other words, the bit cursor leaves a trail of black or white behind it. Selecting Black or White changes the color drawn when the bit cursor is moved. If you want to move the bit cursor without drawing on the screen, select the Move option.

The box cursor is a box displayed on the editing screen, but its size can be changed. It can be as small as a 1 × 1 cell or as large as the entire editing screen. When you're using a utility such as Copy or Flip, the box cursor outlines the area on which the utility will operate. These utilities can be used on a character, on part of a character, on shapes made up of several characters, or on a portion of a character, simply by changing the size of the box. Pressing the Return key when Flip is selected turns the contents of the box cursor upside down, and the Mirror function reverses left and right sides of

the box. The Invert function changes all the white cells inside the box to black cells, and all black cells to white. When Copy is selected, the cursor pad controls a second box cursor, which initially appears on top of the original box. Pressing the Return key copies the contents of the original box to the second box.

You can also use the box cursor to select the 7 × 8 cell character pattern for the Put and Get functions. The character cursor, located in the character set displayed at the bottom of the screen, flashes when the Get or Put function is selected. Use it to select the character that is the source of the Get or the destination of the Put.

The contents of the box cursor are displayed at actual size (one cell = one pixel) in the upper-right corner of the screen. Two parameters, HB and PX, affect how colors are presented. Pressing the Return key when the High Bit menu entry is selected changes the setting of HB. In Apple hi-res graphics, the status of seven one-bit pixels is stored in the lower seven bits of a byte in memory. The eighth bit, the most significant bit, controls the colors in which these bits will be drawn. When drawing on the high-resolution screen in BASIC, the high bit is clear when HCOLOR is between zero and three, and is set when HCOLOR is between four and seven. The display is in blue and orange when the high bit is set, or green and violet when the high bit is clear.

The Even/Odd menu entry controls whether this display starts on an even or an odd pixel (PX). When a shape is shifted by one bit, the colors in the display are reversed (blue for orange or green for violet). The alignment of the shape is changed by pressing Return when Even/Odd is selected.

At the bottom of the screen, all of the characters in a 96-character set are shown. With the RAM/ROM function in the Display menu, the character set displayed can be either the set you are currently editing or the hardware character set in your Apple. Get and Put operate only on the RAM character set, no matter which set is being displayed.

HROUT, the Character Generator

Apple SuperFont uses a machine language graphics utility called HROUT, for high-resolution output. For more information about HROUT, including instructions for using it in your own programs, see the article "Hi-Res Character Graphics for the Apple II," found elsewhere in this book.

Program 1. Apple SuperFont for DOS 3.3 (APPLEFONT)

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
9A 100 IF PEEK (978) < > 157 THEN PRINT "DOS 3.3 NOT
       FOUND": END
F4 110 HGR
56 120 D$ = CHR$ (4)
54 130 PRINT D$;"BLOAD HROUT"
63 140 PRINT D$;"BLOAD NORMAL.SET,A$8D00"
5D 150 POKE 6,0: POKE 7,141
BE 160 POKE 54,0: POKE 55,3: CALL 1002
4C 170 PRINT D$;"BLOAD APPLEFONT2":CE = 4096
CF 180 ONERR GOTO 280
96 190 CALL CE
C4 200 NA$ = "":I = 14 * 256
CD 210 IF PEEK (I) = 141 THEN 260
BE 220 NA$ = NA$ + CHR$ ( PEEK (I)):I = I + 1: IF PE
      EK (I) < > 141 THEN 220
A4 230 IF PEEK (14 * 256 + 32) THEN 250
B8 240 PRINT D$;"BLOAD";NA$;",A$8A00": GOTO 260
E8 250 PRINT D$;"BSAVE";NA$;",A$8A00,L$300"
4B 260 GOSUB 320
F9 270 CALL CE + 3: GOTO 200
F7 280 GOSUB 320: VTAB 18: HTAB 1:EN = PEEK (222)
C8 290 IF EN = 6 OR EN = 7 THEN PRINT "COULDN'T FIND
       "NA$: GOTO 270
0E 300 IF EN = 13 THEN PRINT NA$" ISN'T A CHARACTER
      SET": GOTO 270
B8 310 PRINT "DISK ERROR": GOTO 270
93 320 VTAB 18: HTAB 1: FOR I = 1 TO 80: PRINT " ";:
       NEXT : RETURN
```

Program 2. Apple SuperFont ProDOS Modifications

```
9A 100 IF PEEK (978) < > 190 THEN PRINT "PRODOS NOT
      FOUND": END
AE 160 PRINT D$;"PR# A$300"
```

Program 3. Apple SuperFont Editor (APPLEFONT2)

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter the following three programs.

```
START ADDRESS: 1000
END ADDRESS:   1FDF

1000: 4C 53 12 4C 6D 12 41 00 58
1008: 00 05 05 07 08 00 00 00 BA
```

```
1010: 00 07 08 00 01 00 00 00 FA
1018: 04 03 05 06 06 00 1A 50 B0
1020: 82 00 01 00 00 00 00 00 A1
1028: 00 00 00 00 00 00 00 04 4C
1030: 00 00 00 A9 00 85 1C A9 E2
1038: 20 85 E6 20 F6 F3 A9 02 85
1040: 20 09 1D A0 00 A2 00 18 03
1048: 20 00 1D A2 DC 20 03 1D D0
1050: C8 C8 C8 C8 C0 84 90 ED D3
1058: A2 00 A0 00 18 20 00 1D 3C
1060: A0 B0 20 06 1D E8 E8 E8 9C
1068: E8 E0 E0 90 ED 60 A9 FF 9E
1070: 85 32 A9 8A 85 07 A9 A0 F9
1078: 8D 23 10 A0 15 98 20 5B DA
1080: FB A2 00 AD 23 10 86 24 AC
1088: 20 ED FD EE 23 10 E8 E0 EE
1090: 20 D0 F0 C8 C0 18 D0 E5 8D
1098: A9 8D 85 07 60 20 56 1C 5E
10A0: B9 00 0C CE 2D 10 30 05 1B
10A8: 1D BA 1C D0 03 3D B2 1C 25
10B0: 99 00 0C EE 2D 10 AD 2D 40
10B8: 10 F0 02 A9 03 20 09 1D BF
10C0: 18 AD 2A 10 0A 0A AA E8 55
10C8: AD 2C 10 0A 0A A8 C8 18 0A
10D0: 8A 20 02 11 20 02 11 20 DA
10D8: 02 11 AD 2A 10 CD 09 10 70
10E0: 90 1F ED 0B 10 90 05 CD 1A
10E8: 09 10 B0 15 AD 2C 10 CD 05
10F0: 0A 10 90 0D ED 0C 10 90 4D
10F8: 05 CD 0A 10 B0 03 20 2B 4E
1100: 11 60 20 00 1D E8 E8 20 45
1108: 03 1D AA C8 60 A9 00 A8 27
1110: 99 00 0C C8 D0 FA 20 56 16
1118: 11 60 20 56 1C B9 00 0C 18
1120: 3D BA 1C F0 02 A9 01 8D 68
1128: 2D 10 60 AD 2D 10 F0 02 59
1130: A9 03 18 6D 2F 10 20 09 C4
1138: 1D AD 2C 10 38 ED 0A 10 78
1140: A8 A9 E0 38 ED 09 10 18 8C
1148: 6D 2E 10 6D 2A 10 AA 20 8C
1150: 00 1D 20 03 1D 60 AC 0A BB
1158: 10 8C 2C 10 A0 00 8C 23 6D
1160: 10 AE 09 10 8E 2A 10 A2 38
1168: 00 8E 24 10 20 1A 11 20 5F
1170: 2B 11 EE 2A 10 EE 24 10 81
1178: AE 24 10 EC 0B 10 D0 EC F2
1180: EE 2C 10 EE 23 10 AC 23 EB
1188: 10 CC 0C 10 D0 D3 60 AD AC
1190: 27 10 F0 50 30 28 AD 09 F3
1198: 10 18 6D 0B 10 E9 00 8D DC
```

```
11A0: 2A 10 AD 0A 10 8D 2C 10 51
11A8: AC 0C 10 8C 23 10 20 1A A2
11B0: 11 20 2B 11 EE 2C 10 CE F0
11B8: 23 10 D0 F2 F0 26 AD 09 3E
11C0: 10 18 6D 0B 10 8D 2A 10 6A
11C8: AD 0A 10 8D 2C 10 AC 0C 26
11D0: 10 8C 23 10 A9 00 8D 2D 19
11D8: 10 20 2B 11 EE 2C 10 CE 98
11E0: 23 10 10 F5 AD 28 10 F0 19
11E8: 4F 30 27 AD 0A 10 18 6D AC
11F0: 0C 10 E9 00 8D 2C 10 AD 45
11F8: 09 10 8D 2A 10 AC 0B 10 51
1200: 8C 23 10 20 1A 11 20 2B B7
1208: 11 EE 2A 10 CE 23 10 D0 AA
1210: F2 60 AD 0A 10 18 6D 0C E3
1218: 10 8D 2C 10 AD 09 10 8D 6D
1220: 2A 10 AC 0B 10 8C 23 10 AC
1228: A9 00 8D 2D 10 20 2B 11 0E
1230: EE 2A 10 CE 23 10 10 F5 B4
1238: 60 AD 2F 10 20 09 1D A0 DE
1240: 00 A2 E0 18 20 00 1D A2 88
1248: 17 38 20 03 1D C8 C0 20 E7
1250: 90 EF 60 20 58 FC 20 33 F0
1258: 10 20 1A 19 2C 52 C0 20 AD
1260: 0D 11 A9 C1 8D 06 10 20 65
1268: 9E 1A 20 DA 1B 20 6E 10 5A
1270: A9 00 8D 13 10 8D 27 10 61
1278: 8D 28 10 20 7D 19 AD 13 30
1280: 10 49 02 8D 13 10 A2 00 36
1288: A0 80 AD 00 C0 30 08 E8 92
1290: D0 F8 C8 D0 F5 10 E4 48 83
1298: A0 02 8C 13 10 20 7D 19 65
12A0: 68 2C 10 C0 C9 A0 D0 1B 9F
12A8: AD 21 10 38 69 00 CD 18 70
12B0: 10 D0 02 A9 00 8D 21 10 74
12B8: A9 01 8D 22 10 20 9E 1A 1E
12C0: 4C 01 13 C9 88 D0 17 AD AD
12C8: 22 10 18 E9 00 D0 06 AC 9F
12D0: 21 10 B9 19 10 8D 22 10 5D
12D8: 20 9E 1A 4C 01 13 C9 95 3A
12E0: D0 1F AD 22 10 38 69 00 41
12E8: 8D 22 10 AC 21 10 B9 19 FE
12F0: 10 CD 22 10 B0 05 A9 01 C3
12F8: 8D 22 10 20 9E 1A 4C 01 67
1300: 13 C9 D7 D0 09 CE 27 10 0C
1308: CE 28 10 4C 5A 13 C9 C5 DE
1310: D0 06 CE 28 10 4C 5A 13 F5
1318: C9 D2 D0 09 CE 28 10 EE A8
1320: 27 10 4C 5A 13 C9 D3 D0 45
1328: 06 CE 27 10 4C 5A 13 C9 A6
```

```
1330: C6 D0 06 EE 27 10 4C 5A 0A
1338: 13 C9 D8 D0 09 CE 27 10 64
1340: EE 28 10 4C 5A 13 C9 C3 25
1348: D0 06 EE 28 10 4C 5A 13 32
1350: C9 D6 D0 06 EE 28 10 EE B2
1358: 27 10 AE 21 10 D0 03 4C 14
1360: 75 13 CA D0 03 4C D5 13 74
1368: CA D0 03 4C C5 15 CA D0 36
1370: 03 4C 37 18 60 48 20 31 29
1378: 19 A9 01 8D 14 10 68 C9 0A
1380: C4 F0 08 AD 27 10 0D 28 DC
1388: 10 F0 47 AD 22 10 C9 03 9E
1390: F0 14 69 FF 8D 2D 10 AD 50
1398: 07 10 8D 2A 10 AD 08 10 F1
13A0: 8D 2C 10 20 9D 10 AD 07 2C
13A8: 10 18 6D 27 10 C9 FF D0 75
13B0: 02 A9 36 C9 37 D0 02 A9 50
13B8: 00 8D 07 10 AD 08 10 18 E9
13C0: 6D 28 10 C9 FF D0 02 A9 37
13C8: 1F C9 20 D0 02 A9 00 8D 46
13D0: 08 10 4C 70 12 48 AE 22 C0
13D8: 10 CA D0 03 4C F8 13 CA 3B
13E0: D0 03 4C 74 14 CA D0 03 71
13E8: 4C DA 14 CA D0 03 4C 55 9B
13F0: 15 CA D0 03 4C 64 15 00 BC
13F8: 20 31 19 A9 01 8D 15 10 B1
1400: 20 3F 19 68 4C 70 12 20 1A
1408: 31 19 A9 01 8D 16 10 AD E6
1410: 0B 10 CD 11 10 D0 08 AD 0E
1418: 0C 10 CD 12 10 F0 2A AD 6B
1420: 11 10 8D 0B 10 AD 12 10 A2
1428: 8D 0C 10 20 39 12 20 56 C6
1430: 11 AD 09 10 C9 31 90 05 A7
1438: A9 30 8D 09 10 AD 0A 10 DE
1440: C9 19 90 05 A9 18 8D 0A C8
1448: 10 AD 27 10 0D 28 10 F0 E3
1450: 22 AD 28 10 F0 05 0A 0A B4
1458: 0A 0A 0A 18 6D 27 10 18 0B
1460: 6D 06 10 C9 A0 10 02 69 12
1468: 60 C9 00 30 03 38 E9 60 63
1470: 8D 06 10 60 20 07 14 68 96
1478: C9 8D D0 5B 20 38 15 AD 72
1480: 0A 10 8D 2C 10 AD 0C 10 85
1488: 8D 24 10 A9 00 8D 31 10 C5
1490: AD 09 10 8D 2A 10 AD 0B A4
1498: 10 8D 23 10 A9 00 8D 32 2C
14A0: 10 20 1A 11 AD 2D 10 F0 60
14A8: 02 38 24 18 6E 32 10 EE 31
14B0: 2A 10 CE 23 10 D0 EA AD 45
14B8: 2F 10 F0 02 A9 80 6E 32 19
```

```
14C0: 10 0D 32 10 AC 31 10 91 57
14C8: 1A EE 2C 10 EE 31 10 CE 6B
14D0: 24 10 D0 BC 20 6E 10 4C 1C
14D8: 70 12 20 07 14 68 C9 8D 95
14E0: D0 53 20 38 15 AD 0A 10 51
14E8: 8D 2C 10 AD 0C 10 8D 24 9F
14F0: 10 A9 00 8D 31 10 AD 09 92
14F8: 10 8D 2A 10 AD 0B 10 8D 1A
1500: 23 10 AC 31 10 B1 1A 8D 71
1508: 32 10 4E 32 10 A9 00 69 CC
1510: 00 8D 2D 10 20 9D 10 EE CA
1518: 2A 10 CE 23 10 D0 EB AD B0
1520: 32 10 0A 0A 8D 2F 10 EE 81
1528: 2C 10 EE 31 10 CE 24 10 71
1530: D0 C4 20 DA 1B 4C 70 12 A2
1538: AD 06 10 38 E9 A0 85 1A 37
1540: A9 00 85 1B A2 03 06 1A E8
1548: 26 1B CA D0 F9 A5 1B 18 67
1550: 69 8A 85 1B 60 20 31 19 33
1558: 68 C9 8D D0 04 A9 00 F0 9F
1560: 12 4C 70 12 20 31 19 68 36
1568: C9 8D D0 04 A9 01 D0 03 2B
1570: 4C 70 12 8D 20 0E A9 A0 25
1578: A2 1F 9D 00 0E CA 10 FA 26
1580: A9 FF 85 32 A9 11 20 5B 80
1588: FB A9 00 85 24 A0 00 B9 D0
1590: A7 15 F0 06 20 ED FD C8 CF
1598: D0 F5 20 6A FD BD 00 02 3C
15A0: 9D 00 0E CA 10 F7 60 C5 EE
15A8: CE D4 C5 D2 A0 CE C1 CD E6
15B0: C5 A0 CF C6 A0 C3 C8 C1 B3
15B8: D2 C1 C3 D4 C5 D2 A0 D3 11
15C0: C5 D4 BA 8D 00 48 AE 22 D3
15C8: 10 CA D0 03 4C F8 13 CA 2F
15D0: D0 03 4C EE 15 CA D0 03 15
15D8: 4C 37 16 CA D0 03 4C EB 7D
15E0: 16 CA D0 03 4C 66 17 CA 08
15E8: D0 03 4C E1 17 00 20 31 0E
15F0: 19 A9 01 8D 15 10 68 AD 72
15F8: 27 10 0D 28 10 F0 35 AD 3B
1600: 27 10 18 6D 0B 10 D0 02 D9
1608: A9 01 8D 0B 10 18 6D 09 70
1610: 10 C9 38 D0 03 CE 0B 10 44
1618: AD 28 10 18 6D 0C 10 D0 35
1620: 02 A9 01 8D 0C 10 18 6D EE
1628: 0A 10 C9 21 D0 03 CE 0C E4
1630: 10 20 8F 11 4C 70 12 20 D7
1638: 31 19 A9 01 8D 17 10 AD 1F
1640: 27 10 0D 28 10 F0 32 AD 7E
1648: 27 10 18 6D 0D 10 10 02 B0
```

```
1650: A9 00 8D 0D 10 18 6D 0F 9E
1658: 10 C9 38 D0 03 CE 0D 10 90
1660: AD 28 10 18 6D 0E 10 10 C4
1668: 02 A9 00 8D 0E 10 18 6D 27
1670: 10 10 C9 21 D0 03 CE 0E 32
1678: 10 68 C9 8D D0 6A 20 70 B9
1680: 1C AD 0A 10 8D 23 10 AD 2F
1688: 0E 10 8D 25 10 AD 09 10 1D
1690: 8D 24 10 AD 0D 10 8D 26 53
1698: 10 AD 23 10 8D 2C 10 AD 88
16A0: 24 10 8D 2A 10 20 56 1C 01
16A8: B9 00 0D 3D BA 1C F0 02 51
16B0: A9 01 8D 2D 10 AD 25 10 08
16B8: 8D 2C 10 AD 26 10 8D 2A 4A
16C0: 10 20 9D 10 EE 24 10 EE C8
16C8: 26 10 AD 09 10 18 6D 0B 19
16D0: 10 CD 24 10 D0 C3 EE 23 94
16D8: 10 EE 25 10 AD 0A 10 18 3C
16E0: 6D 0C 10 CD 23 10 D0 A5 46
16E8: 4C 70 12 20 31 19 A9 01 DD
16F0: 8D 15 10 20 3F 19 68 C9 26
16F8: 8D D0 68 20 70 1C AD 0A 88
1700: 10 8D 23 10 8D 25 10 AD CD
1708: 09 10 8D 24 10 18 6D 0B 79
1710: 10 E9 00 8D 26 10 AD 23 89
1718: 10 8D 2C 10 AD 24 10 8D E3
1720: 2A 10 20 56 1C B9 00 0D A5
1728: 3D BA 1C F0 02 A9 01 8D 7C
1730: 2D 10 AD 25 10 8D 2C 10 20
1738: AD 26 10 8D 2A 10 20 9D 11
1740: 10 EE 24 10 AD 26 10 CD AB
1748: 09 10 F0 05 CE 26 10 B0 4D
1750: C5 EE 23 10 EE 25 10 AD 5C
1758: 0A 10 18 6D 0C 10 CD 25 CA
1760: 10 D0 A4 4C 70 12 20 31 61
1768: 19 A9 01 8D 15 10 20 3F EE
1770: 19 68 C9 8D D0 68 20 70 30
1778: 1C AD 0A 10 8D 23 10 18 93
1780: 6D 0C 10 E9 00 8D 25 10 99
1788: AD 09 10 8D 24 10 8D 26 4D
1790: 10 AD 23 10 8D 2C 10 AD 82
1798: 24 10 8D 2A 10 20 56 1C FA
17A0: B9 00 0D 3D BA 1C F0 02 4B
17A8: A9 01 8D 2D 10 AD 25 10 02
17B0: 8D 2C 10 AD 26 10 8D 2A 44
17B8: 10 20 9D 10 EE 24 10 EE C2
17C0: 26 10 AD 09 10 18 6D 0B 13
17C8: 10 CD 24 10 D0 C3 EE 23 8E
17D0: 10 AD 25 10 CD 0A 10 F0 BF
17D8: 05 CE 25 10 B0 AA 4C 70 1C
```

```
17E0: 12 20 31 19 A9 01 8D 15 59
17E8: 10 20 3F 19 68 C9 8D D0 F6
17F0: 43 20 70 1C AD 0A 10 8D DB
17F8: 2C 10 AD 09 10 8D 2A 10 A2
1800: 20 56 1C B9 00 0D 3D BA 5E
1808: 1C D0 03 A9 01 2C A9 00 81
1810: 8D 2D 10 20 9D 10 EE 2A 8B
1818: 10 AD 09 10 18 6D 0B 10 7A
1820: CD 2A 10 D0 DB EE 2C 10 D3
1828: AD 0A 10 18 6D 0C 10 CD BE
1830: 2C 10 D0 C6 4C 70 12 48 91
1838: 20 31 19 AE 22 10 CA D0 8A
1840: 03 4C 63 18 CA D0 03 4C DE
1848: 7A 18 CA D0 03 4C 94 18 AC
1850: CA D0 03 4C B2 18 CA D0 9B
1858: 03 4C F2 18 CA D0 03 4C E8
1860: 0C 19 00 68 C9 8D D0 0F 98
1868: A9 04 38 ED 2F 10 8D 2F 58
1870: 10 20 56 11 20 DA 1B 4C 7B
1878: 70 12 68 C9 8D D0 12 A9 8C
1880: 01 38 ED 2E 10 8D 2E 10 03
1888: 20 39 12 20 56 11 20 DA 6D
1890: 1B 4C 70 12 68 C9 8D D0 E6
1898: 16 A9 01 38 ED 30 10 8D BF
18A0: 30 10 F0 05 2C 53 C0 B0 3C
18A8: 06 2C 52 C0 20 DA 1B 4C 2C
18B0: 70 12 A9 01 8D 15 10 20 E3
18B8: 3F 19 68 C9 8D D0 30 A9 32
18C0: 00 8D 2D 10 AD 0A 10 8D 3E
18C8: 2C 10 AD 0C 10 8D 23 10 96
18D0: AD 09 10 8D 2A 10 AD 0B EC
18D8: 10 8D 24 10 20 9D 10 EE 80
18E0: 2A 10 CE 24 10 D0 F5 EE E4
18E8: 2C 10 CE 23 10 D0 E1 4C 13
18F0: 70 12 68 C9 8D D0 12 20 7B
18F8: 58 FC 20 33 10 20 9E 1A 24
1900: 20 6E 10 20 DA 1B 20 0D 72
1908: 11 4C 70 12 68 C9 8D D0 5B
1910: 06 20 1A 19 20 6E 10 4C 49
1918: 70 12 A9 8A 85 1B A9 00 D0
1920: 85 1A A2 03 A0 00 91 1A 62
1928: C8 D0 FB E6 1B CA D0 F6 7D
1930: 60 48 A2 00 8A 9D 14 10 FB
1938: E8 E0 04 90 F8 68 60 AD 78
1940: 27 10 0D 28 10 F0 35 AD 8A
1948: 27 10 18 6D 09 10 10 02 96
1950: A9 00 8D 09 10 18 6D 0B 60
1958: 10 C9 38 D0 03 CE 09 10 8E
1960: AD 28 10 18 6D 0A 10 10 BA
1968: 02 A9 00 8D 0A 10 18 6D 0D
```

```
1970: 0C 10 C9 21 D0 03 CE 0A 32
1978: 10 20 56 11 60 A9 00 20 60
1980: 09 1D AD 14 10 F0 06 AD 73
1988: 13 10 20 09 1D AD 07 10 9A
1990: 0A 0A AA AD 08 10 0A 0A 19
1998: A8 18 20 00 1D BA 69 04 13
19A0: AA 20 03 1D 98 69 04 A8 7D
19A8: 20 06 1D BA 38 E9 04 18 42
19B0: AA 20 03 1D 98 38 E9 04 EF
19B8: A8 20 06 1D A9 00 20 09 70
19C0: 1D AD 15 10 F0 06 AD 13 9E
19C8: 10 20 09 1D AD 09 10 0A B9
19D0: 0A AA AD 0A 10 0A 0A A8 6E
19D8: 18 20 00 1D AD 09 10 6D 10
19E0: 0B 10 0A 0A AA 20 03 1D 77
19E8: AD 0A 10 6D 0C 10 0A 0A 0C
19F0: A8 20 06 1D AD 09 10 0A CD
19F8: 0A AA 20 03 1D AD 0A 10 D2
1A00: 0A 0A A8 20 06 1D A9 3F 0A
1A08: 85 32 A9 BA 85 07 AD 06 13
1A10: 10 29 1F 85 24 AD 06 10 C6
1A18: 29 60 A2 05 4A CA D0 FC B9
1A20: 69 14 20 5B FB AD 16 10 9A
1A28: F0 09 AD 13 10 F0 04 A9 F3
1A30: FF 85 32 AD 06 10 20 ED 85
1A38: FD A9 8D 85 07 AD 17 10 0D
1A40: F0 43 AD 13 10 20 09 1D D4
1A48: AD 0D 10 0A 0A AA AD 0E 9D
1A50: 10 0A 0A A8 18 20 00 1D 39
1A58: AD 0D 10 6D 0F 10 0A 0A 56
1A60: AA 20 03 1D AD 0E 10 6D 57
1A68: 10 10 0A 0A A8 20 06 1D 79
1A70: AD 0D 10 0A 0A AA 20 03 9F
1A78: 1D AD 0E 10 0A 0A A8 20 53
1A80: 06 1D 4C 9D 1A AD 09 10 0C
1A88: 8D 0D 10 AD 0A 10 8D 0E 5D
1A90: 10 AD 0B 10 8D 0F 10 AD 11
1A98: 0C 10 8D 10 10 60 A9 08 E6
1AA0: 20 5B FB A9 3F 85 32 AC F6
1AA8: 21 10 B9 19 10 8D 23 10 47
1AB0: 18 69 01 38 ED 22 10 8D 94
1AB8: 24 10 B9 1D 10 A8 20 FF 6F
1AC0: 1A A9 8D 20 ED FD A9 FF DA
1AC8: 85 32 AD 24 10 CD 23 10 52
1AD0: D0 04 A9 3F 85 32 20 FF CC
1AD8: 1A CE 23 10 D0 E8 A9 FF B0
1AE0: 85 32 AC 21 10 AD 19 10 85
1AE8: 8D 23 10 A2 08 A9 20 85 85
1AF0: 24 20 11 1B EE 23 10 A9 E0
1AF8: 08 CD 23 10 D0 ED 60 A2 AB
```

```
1B00: 08 A9 20 85 24 B9 1F 1B 62
1B08: F0 07 20 ED FD CA C8 D0 D8
1B10: F4 C8 E0 00 F0 08 A9 A0 AA
1B18: 20 ED FD CA D0 FA 60 C2 3C
1B20: C9 D4 A0 C5 C4 C9 D4 00 D7
1B28: C2 CC C1 C3 CB 00 D7 C8 3E
1B30: C9 D4 C5 00 CD CF D6 C5 5A
1B38: 00 C3 C8 C1 D2 D3 C5 D4 DA
1B40: D3 00 CD CF D6 C5 A0 C2 E8
1B48: CF D8 00 D0 D5 D4 A0 C3 B0
1B50: C8 C1 D2 00 C7 C5 D4 A0 55
1B58: C3 C8 C1 D2 00 CC CF C1 9C
1B60: C4 A0 D3 C5 D4 00 D3 C1 08
1B68: D6 C5 A0 D3 C5 D4 00 D5 24
1B70: D4 C9 CC C9 D4 D9 00 CD 95
1B78: CF D6 C5 A0 C2 CF D8 00 16
1B80: C2 CF D8 A0 D3 C9 DA C5 72
1B88: 00 C3 CF D0 D9 00 CD C9 EA
1B90: D2 D2 CF D2 00 C6 CC C9 8A
1B98: D0 00 C9 CE D6 C5 D2 D4 A5
1BA0: 00 C4 C9 D3 D0 CC C1 D9 95
1BA8: 00 C8 C9 A0 C2 C9 D4 00 3B
1BB0: C5 D6 C5 CE AF CF C4 C4 30
1BB8: 00 D2 C1 CD AF D2 CF CD EE
1BC0: 00 C3 CC D2 A0 C2 CF D8 37
1BC8: 00 C3 CC D2 A0 D3 C3 D2 65
1BD0: CE 00 C3 CC D2 A0 D3 C5 3A
1BD8: D4 00 A9 05 20 5B FB A9 0F
1BE0: 20 85 24 A9 FF 85 32 A0 C2
1BE8: 00 A9 04 20 2C 1C AD 2F 68
1BF0: 10 D0 09 A0 04 A9 03 20 7B
1BF8: 2C 1C F0 07 A0 07 A9 03 52
1C00: 20 2C 1C A9 8D 20 ED FD 38
1C08: A9 20 85 24 A0 0A A9 04 94
1C10: 20 2C 1C AD 2E 10 D0 09 1E
1C18: A0 0E A9 04 20 2C 1C F0 74
1C20: 0A A0 12 A9 04 20 2C 1C 77
1C28: AD 2E 10 60 8D 29 10 A2 9E
1C30: 00 B9 3F 1C 20 ED FD C8 FD
1C38: E8 EC 29 10 D0 F3 60 C8 26
1C40: C2 BA A0 C3 CC D2 D3 C5 F7
1C48: D4 D0 D8 BA A0 C5 D6 C5 75
1C50: CE CF C4 C4 A0 00 AD 2A 53
1C58: 10 0A 0A 8D 2B 10 AD 2C 56
1C60: 10 29 07 AA AD 2C 10 4A FE
1C68: 4A 4A 18 6D 2B 10 A8 60 7D
1C70: AD 0A 10 8D 2C 10 AD 0C E5
1C78: 10 8D 23 10 AD 09 10 8D C0
1C80: 2A 10 AD 0B 10 8D 24 10 47
1C88: 20 1A 11 B9 00 0D CE 2D 14
```

```
1C90: 10 30 05 1D BA 1C D0 03 3A
1C98: 3D B2 1C 99 00 0D EE 2D 78
1CA0: 10 EE 2A 10 CE 24 10 D0 DA
1CA8: DF EE 2C 10 CE 23 10 D0 07
1CB0: CB 60 FE FD FB F7 EF DF 26
1CB8: BF 7F 01 02 04 08 10 20 71
1CC0: 40 80 8D A0 A0 A0 A0 A0 5E
1CC8: A0 A0 A0 A0 A0 A0 A0 A0 01
1CD0: A0 A0 A0 A0 A0 A0 A0 A0 09
1CD8: A0 A0 A0 A0 A0 A0 A0 A0 11
1CE0: A0 A0 00 FF 00 FF 00 FF 91
1CE8: 00 FF B7 FF 00 FF 00 FF 18
1CF0: 00 FF 00 FF 00 FF 00 FF 29
1CF8: 00 FF 00 FF 00 FF 00 FF 31
1D00: 4C 05 1F 4C 22 1E 4C C4 31
1D08: 1D 4C F6 1E 00 00 00 00 A4
1D10: 00 00 00 00 00 A5 1C 51 6A
1D18: 26 25 30 51 26 91 26 60 ED
1D20: 85 45 86 46 84 47 60 A5 4B
1D28: 45 A6 46 A4 47 60 A5 1C E4
1D30: 4A 4A 4A 4C 40 1D A5 1C 0E
1D38: 4A 4C 40 1D A5 1C 4A 4A 01
1D40: 29 0F A8 B9 4F 1D 24 1C D6
1D48: 10 02 09 80 85 1C 60 00 91
1D50: 11 22 33 44 55 66 77 08 81
1D58: 19 2A 3B 4C 5D 6E 7F 00 79
1D60: 04 08 0C 10 14 18 1C 00 5A
1D68: 04 08 0C 10 14 18 1C 01 63
1D70: 05 09 0D 11 15 19 1D 01 6A
1D78: 05 09 0D 11 15 19 1D 02 73
1D80: 06 0A 0E 12 16 1A 1E 02 7A
1D88: 06 0A 0E 12 16 1A 1E 03 83
1D90: 07 0B 0F 13 17 1B 1F 03 8A
1D98: 07 0B 0F 13 17 1B 1F 81 11
1DA0: 82 84 88 90 A0 C0 81 83 E5
1DA8: 87 8F 9F BF FF FF FE FC 75
1DB0: F8 F0 E0 C0 00 2A 55 7F 9E
1DB8: 80 AA D5 FF 22 11 77 5D 3A
1DC0: A2 91 F7 BB 08 20 20 1D 89
1DC8: C0 C0 90 03 4C B8 1F AC 06
1DD0: 0D 1D B9 9F 1D 85 30 A5 0F
1DD8: 27 29 1F 05 E6 85 27 A5 66
1DE0: 47 38 ED 10 1D AA 6E 11 0D
1DE8: 1D 10 17 E8 AC 0C 1D 20 17
1DF0: 15 1D CA F0 23 AD 11 1D 74
1DF8: 18 20 D3 F4 20 3C 1D 4C 89
1E00: EC 1D CA AC 0C 1D 20 15 48
1E08: 1D E8 F0 0C AD 11 1D 20 F7
1E10: D3 F4 20 3C 1D 4C 03 1E 79
1E18: A5 47 8D 10 1D 20 27 1D 80
```

```
1E20: 28 60 08 20 20 1D 90 07 29
1E28: E0 18 90 03 4C B8 1F A0 41
1E30: 00 8C 0E 1D 8E 0F 1D 28 36
1E38: 08 90 03 EE 0E 1D A5 27 43
1E40: 29 1F 05 E6 85 27 AD 0F 1B
1E48: 1D A2 E0 8E 12 1D AE 0E 31
1E50: 1D 8E 13 1D 4E 13 1D A2 8E
1E58: 06 90 03 69 1F 38 2E 13 FB
1E60: 1D CD 12 1D 90 06 EE 13 40
1E68: 1D ED 12 1D 4E 12 1D CA 82
1E70: D0 EC 8D 14 1D AE 0D 1D 1E
1E78: AC 0C 1D CC 13 1D D0 1E 4B
1E80: EC 14 1D 90 0B BD A6 1D 9E
1E88: AE 14 1D 3D AD 1D B0 09 E4
1E90: BD AD 1D AE 14 1D 3D A6 DB
1E98: 1D 85 30 4C EB 1E 90 27 AF
1EA0: BD A6 1D 85 30 20 15 1D AA
1EA8: 20 36 1D CE 0C 1D AC 0C 4D
1EB0: 1D CC 13 1D F0 06 A5 1C E9
1EB8: 91 26 B0 EC AE 14 1D BD E9
1EC0: AD 1D 85 30 4C EB 1E BD DA
1EC8: AD 1D 85 30 20 15 1D 20 86
1ED0: 2E 1D EE 0C 1D AC 0C 1D DA
1ED8: CC 13 1D F0 06 A5 1C 91 83
1EE0: 26 90 EC AE 14 1D BD A6 14
1EE8: 1D 85 30 20 15 1D 8E 0D 64
1EF0: 1D 20 27 1D 28 60 08 20 6D
1EF8: 20 1D 29 0F A8 B9 B4 1D 55
1F00: 85 1C 4C 9B 1F 08 20 20 C4
1F08: 1D 90 07 E0 18 90 03 4C 3D
1F10: B8 1F C0 C0 90 03 20 B8 20
1F18: 1F A9 01 2D 10 1D F0 03 1D
1F20: 20 3C 1D A9 03 2D 0C 1D BD
1F28: F0 07 AA 20 36 1D CA D0 84
1F30: FA A5 47 8D 10 1D A5 46 9D
1F38: 8D 0F 1D A0 00 8C 0E 1D 1A
1F40: 28 08 90 03 EE 0E 1D 2C EC
1F48: 10 1D 10 02 A0 50 50 02 E0
1F50: A0 28 84 26 A9 08 2D 10 B3
1F58: 1D F0 06 A9 80 05 26 85 A6
1F60: 26 AD 10 1D 29 3F A8 B9 42
1F68: 5F 1D 85 27 AD 0F 1D A2 47
1F70: E0 8E 12 1D AE 0E 1D 8E 4D
1F78: 0C 1D 4E 0C 1D A2 06 90 9E
1F80: 03 69 1F 38 2E 0C 1D CD AB
1F88: 12 1D 90 06 EE 0C 1D ED 59
1F90: 12 1D 4E 12 1D CA D0 EC AC
1F98: 8D 0D 1D A9 01 2D 10 1D 19
1FA0: F0 03 20 3C 1D A9 03 2D A2
1FA8: 0C 1D F0 07 AA 20 2E 1D 12
```

```
1FB0: CA D0 FA 20 27 1D 28 60 48
1FB8: A0 00 B9 C6 1F 20 ED FD 3E
1FC0: C8 C0 19 D0 F5 00 D3 C3 DE
1FC8: D2 C5 C5 CE A0 C2 CF D5 0D
1FD0: CE C4 C1 D2 D9 A0 C5 D8 C2
1FD8: C3 C5 C5 C4 C5 C4 8D 00 CB
```

Program 4. Apple SuperFont NORMAL.SET

```
START ADDRESS: 8D00
END ADDRESS:   8FFF

8D00: 00 00 00 00 00 00 00 00 1B
8D08: 08 08 08 08 08 00 08 00 FA
8D10: 14 14 14 00 00 00 00 00 BC
8D18: 14 14 3E 14 3E 14 14 00 B5
8D20: 08 3C 0A 1C 28 1E 08 00 1B
8D28: 06 26 10 08 04 32 30 00 9B
8D30: 04 0A 0A 04 2A 12 2C 00 43
8D38: 08 08 08 00 00 00 00 00 5A
8D40: 08 04 02 02 02 04 08 00 F0
8D48: 08 10 20 20 20 10 08 00 C2
8D50: 08 2A 1C 08 1C 2A 08 00 97
8D58: 00 08 08 3E 08 08 00 00 BA
8D60: 00 00 00 00 00 08 08 04 00 E3
8D68: 00 00 00 3E 00 00 00 00 67
8D70: 00 00 00 00 00 00 08 00 9B
8D78: 00 20 10 08 04 02 00 00 46
8D80: 1C 22 32 2A 26 22 1C 00 0D
8D88: 08 0C 08 08 08 08 1C 00 C4
8D90: 1C 22 20 18 04 02 3E 00 6C
8D98: 3E 20 10 18 20 22 1C 00 20
8DA0: 10 18 14 12 3E 10 10 00 BF
8DA8: 3E 02 1E 20 20 22 1C 00 EA
8DB0: 38 04 02 1E 22 22 1C 00 DC
8DB8: 3E 20 10 08 04 04 04 00 B5
8DC0: 1C 22 22 1C 22 22 1C 00 4A
8DC8: 1C 22 22 3C 20 10 0E 00 DF
8DD0: 00 00 08 00 08 00 00 00 2D
8DD8: 00 00 08 00 08 08 04 00 5D
8DE0: 10 08 04 02 04 08 10 00 07
8DE8: 00 00 3E 00 3E 00 00 00 BD
8DF0: 04 08 10 20 10 08 04 00 BC
8DF8: 1C 22 10 08 08 00 08 00 7D
8E00: 1C 22 2A 3A 1A 02 3C 00 ED
8E08: 08 14 22 22 3E 22 22 00 53
8E10: 1E 22 22 1E 22 22 1E 00 C0
8E18: 1C 22 02 02 02 22 1C 00 FC
8E20: 1E 22 22 22 22 22 1E 00 11
8E28: 3E 02 02 1E 02 02 3E 00 9B
8E30: 3E 02 02 1E 02 02 02 00 2B
8E38: 3C 02 02 02 32 22 3C 00 E6
8E40: 22 22 22 3E 22 22 22 00 FC
8E48: 1C 08 08 08 08 08 1C 00 8F
8E50: 20 20 20 20 20 22 1C 00 4D
8E58: 22 12 0A 06 0A 12 22 00 89
8E60: 02 02 02 02 02 02 3E 00 F3
8E68: 22 36 2A 2A 22 22 22 00 E9
```

```
8E70: 22 22 26 2A 32 22 22 00 EC
8E78: 1C 22 22 22 22 22 1C 00 64
8E80: 1E 22 22 1E 02 02 02 00 77
8E88: 1C 22 22 22 2A 12 2C 00 94
8E90: 1E 22 22 1E 0A 12 22 00 48
8E98: 1C 22 02 1C 20 22 1C 00 10
8EA0: 3E 08 08 08 08 08 08 00 D0
8EA8: 22 22 22 22 22 22 1C 00 97
8EB0: 22 22 22 22 22 14 08 00 3F
8EB8: 22 22 22 2A 2A 36 22 00 C4
8EC0: 22 22 14 08 14 22 22 00 E7
8EC8: 22 22 14 08 08 08 08 00 F2
8ED0: 3E 20 10 08 04 02 3E 00 3C
8ED8: 3E 06 06 06 06 06 3E 00 7C
8EE0: 00 02 04 08 10 20 00 00 80
8EE8: 3E 30 30 30 30 30 3E 00 F8
8EF0: 00 00 08 14 22 00 00 00 61
8EF8: 00 00 00 00 00 00 00 7F 95
8F00: 04 08 10 00 00 00 00 00 25
8F08: 00 00 1C 20 3C 22 3C 00 8F
8F10: 02 02 1E 22 22 22 1E 00 6C
8F18: 00 00 3C 02 02 02 3C 00 6F
8F20: 20 20 3C 22 22 22 3C 00 13
8F28: 00 00 1C 22 3E 02 3C 00 5F
8F30: 18 24 04 1E 04 04 04 00 FE
8F38: 00 00 1C 22 22 3C 20 1C 5B
8F40: 02 02 1E 22 22 22 22 00 A4
8F48: 08 00 0C 08 08 08 1C 00 06
8F50: 10 00 18 10 10 10 12 0C 6C
8F58: 02 02 22 12 0E 12 22 00 5B
8F60: 0C 08 08 08 08 08 1C 00 A1
8F68: 00 00 36 2A 2A 2A 22 00 2F
8F70: 00 00 1E 22 22 22 22 00 53
8F78: 00 00 1C 22 22 22 1C 00 0F
8F80: 00 00 1E 22 22 1E 02 02 15
8F88: 00 00 3C 22 22 3C 20 20 B3
8F90: 00 00 3A 06 02 02 02 00 73
8F98: 00 00 3C 02 1C 20 1E 00 FC
8FA0: 04 04 1E 04 04 24 18 00 A7
8FA8: 00 00 22 22 22 32 2C 00 60
8FB0: 00 00 22 22 22 14 08 00 A7
8FB8: 00 00 22 22 2A 2A 36 00 A4
8FC0: 00 00 22 14 08 14 22 00 3A
8FC8: 00 00 22 22 22 3C 20 1C AC
8FD0: 00 00 3E 10 08 04 3E 00 85
8FD8: 38 0C 0C 06 0C 0C 38 00 F9
8FE0: 08 08 08 08 08 08 08 08 FF
8FE8: 0E 18 18 30 18 18 0E 00 58
8FF0: 2C 1A 00 00 00 00 00 00 AC
8FF8: 00 2A 14 2A 14 2A 00 00 11
```

Program 5. Apple SuperFont HROUT

```
START ADDRESS: 0300
END ADDRESS:   0357

0300: D8 78 85 45 86 46 84 47 33
0308: A6 07 0A 0A B0 04 10 3E F8
0310: 30 04 10 01 E8 E8 0A 86 C6
0318: 1B 18 65 06 85 1A 90 02 76
0320: E6 1B A5 28 85 08 A5 29 58
0328: 29 03 05 E6 85 09 A2 08 30
0330: A0 00 B1 1A 24 32 30 02 AA
0338: 49 7F A4 24 91 08 E6 1A 2E
0340: D0 02 E6 1B A5 09 18 69 A8
0348: 04 85 09 CA D0 E2 A5 45 22
0350: A6 46 A4 47 58 4C F0 FD 18
```

Hi-Res Character Graphics for the Apple II

Tim Victor

"HROUT," a short machine language utility, lets you print text on the Apple II's hi-res screens. For all Apple II–series computers using either DOS 3.3 or ProDOS.

"HROUT" is a machine language graphics utility which will work on any Apple II–series computer. Just as important, it's fully relocatable, so that it can be installed anywhere in RAM. HROUT links into the standard character output vector and permits text to be displayed on either hi-res screen. Because the standard text output routine is also active, the PRINT command, and any other text commands, can be used to create hi-res text. HROUT's only limitation is that it can't perform screen scrolls at the bottom of the screen. HROUT uses the same character sets as "HRCG," the character generator included on the *DOS Tool Kit* disk. Several character sets are on the *Tool Kit* disk, and new sets can be created with "Apple SuperFont," another program included in this book.

Putting HROUT on Disk

To make it easy to enter, HROUT is listed in our "AppleMLX" format. Make sure you have a copy of AppleMLX (Appendix C) on disk and have read the instructions that accompany the article.

Once it's run, AppleMLX will ask you to enter two numbers, a starting and an ending address for HROUT. Your responses should be

```
STARTING ADDRESS? 0300
ENDING ADDRESS?   0357
```

Type in the short listing and save it using the filename HROUT (make sure you save it with this name so that you can use it with the Apple SuperFont program).

In Your Own Programs

To use HROUT in your own programs, BLOAD it into memory. It can be loaded almost anywhere in memory, but to make things simpler, we'll use location $300. (Don't place HROUT in the first 256 bytes of RAM—locations 0–255—the Apple's operating system uses this area to store important data. Changing anything here could easily cause your computer to crash.)

HROUT needs a character set to display, since the character ROM used in text mode can't be read by the CPU. Each character set can contain 96 characters, using eight bytes per character. A set can be created with Apple SuperFont and then BLOADed from disk with

BLOAD *ACHARSET*,A$8D00

This command loads a character set, called *ACHARSET*, at $8D00, near the top of available RAM. If you'll be using HROUT from a BASIC program, you need to protect the character set so that string storage won't overwrite it. The command to do this is

HIMEM: 141*256

(Note: This will cause problems in ProDOS if you try to do any disk operations—it keeps the operating system from setting up a file buffer. You can cancel this before disk I/O with HIMEM: 150*256, then restore it when the disk operations are done.)

You have to let HROUT know which character set to use by POKEing the address of the character set into locations 6 and 7, in a low byte/high byte format. If you put your character set at $8D00 (36096 decimal), the POKEs are

POKE 6,0 : POKE 7,141

(Remember that you POKE the low byte value into location 6. The high byte value, 141 here, is multiplied by 256. 141*256=36096.)

If you're using DOS 3.3, you can activate HROUT by entering

POKE 54,0 : POKE 55,3 : CALL 1002

When in immediate mode, these commands have to be entered together on a multistatement line (separated by colons). They can be on separate lines in a BASIC program, but the three commands should be executed one after another.

Since locations 54 and 55 are being POKEd with the low and high bytes of the address of HROUT, these POKEs will be different if you put HROUT somewhere other than $300.

From ProDOS, it's easier to turn on HROUT. Just type

PR# A$300

Since the last instruction in HROUT is a jump to COUT1, the monitor routine to display a character on the text screen, nothing should appear any different when you're in text mode. To see the hi-res text, type HGR to enable the hi-res screen and HOME to get the cursor out of the four-line text window. You should see whatever you're typing displayed on the screen in a hi-res character set.

In DOS 3.3 HROUT can be canceled with this command sequence:

POKE 54,240 : POKE 55,253 : CALL 1002

To cancel HROUT in ProDOS, enter

PR#0

Cursors and Scrolling

Since HROUT concludes by calling the standard ROM routine for displaying a character on the text screen, all cursor control remains the same. You can move to any location on the screen by using the HTAB and VTAB commands. HOME still moves the cursor to the upper left of the screen, but won't clear the hi-res screen. To get the equivalent of a text HOME, use HOME : CALL −3092. The routine at −3092 clears the current hi-res screen and turns on hi-res graphics.

If you need to know what's where on the screen, you can PEEK to the text screen. By taking a couple of precautions, both text and hi-res screens should be the same. First of all, make sure that you clear both screens at the same time, as mentioned above. Second, don't let the text screen scroll. In order to make HROUT as small (88 bytes) and fast as possible, no provision was made for scrolling the screen. This could even be to your advantage for many applications, but you have to be careful if you want the text and graphics screens to agree.

The biggest problem arises when you print to the last character on the twenty-fourth line. Even if you follow the PRINT statement with a semicolon, the cursor will wrap onto

the twenty-fifth line and the screen will scroll. There *is* a solution: Fool the computer into thinking the screen has 25 lines by using POKE 35,25. The output routine will then have no qualms at all about advancing the cursor to the twenty-fifth line, leaving it there, and even printing there. A lot of responsibility now rests on your shoulders, because the twenty-fifth line doesn't really exist. Printing something there is the same thing as POKEing out of the range of the text screen. That could cause significant problems.

HROUT pays attention to two important BASIC variables: the current hi-res graphics page, zero page location 230 ($E6); and the current text attribute, location 50 ($32). Characters will always be printed to the page set by the last HGR or HGR2 unless you POKE a different value into location 230. Text will be drawn on the same screen as HPLOT, HLINE, and so on.

If you change the text attribute with the INVERSE or FLASH command, the bit patterns will be reversed before they are plotted on the screen, inverting the character. The NORMAL command also works, canceling inverted printing.

Now you can label high-resolution charts and graphs with a choice of any font. You can even design these fonts yourself with the Apple SuperFont Editor. Letters of the alphabet can become detailed shapes, permitting fast high-resolution game graphics in BASIC. In fact, we've started using this technique ourselves for some of the Apple games published by COMPUTE! Publications.

HROUT Zero Page Memory Usage

Hex	Decimal	Usage
$6–7	6–7	Character set address (low byte, high byte). Set by user to point to first byte of character definitions. Never changed by HROUT, but can be changed by user to switch between character sets.
$8–9	8–9	Hi-res row address (low byte, high byte). Updated several times whenever a character is drawn. Doesn't need initialization. Can be changed between calls to HROUT with no ill effect.
$1A–1B	26–27	Character pattern address (low byte, high byte). Also reinitialized with each call.

$24	36	Monitor column position. Contains cursor's current column. Used but not changed by HROUT. It can be changed with the BASIC HTAB command.
$25	37	Monitor row position. Can be changed with VTAB.
$28–29	40–41	BASL,BASH: Monitor's pointer to the start of the current text line. Set by VTAB. Used by HROUT to calculate hi-res row address ($8–9).
$32	50	Monitor text attribute (NORMAL, FLASH, INVERSE). HROUT checks here and inverts text if INVERSE or FLASH is in effect.
$36–37	54–55	Monitor output routine selector (CSWL). Contains address of the standard routine for displaying a character. Normally points to a DOS routine which looks for DOS commands. When this vector is changed and the DOS I/O vectors are reinitialized (CALL 1002), DOS passes its output to HROUT.
$45–47	69–71	Monitor register storage. HROUT stores the registers here upon entry, then restores them when exiting.
$E6	230	BASIC hi-res page switch. Set to 32 ($20) for page one, 64 ($40) for page two. Used but not changed by HROUT.

HROUT

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program.

```
START ADDRESS: 0300
END ADDRESS:   0357

0300: D8 78 85 45 86 46 84 47 33
0308: A6 07 0A 0A B0 04 10 3E F8
0310: 30 04 10 01 E8 E8 0A 86 C6
0318: 1B 18 65 06 85 1A 90 02 76
0320: E6 1B A5 28 85 08 A5 29 58
0328: 29 03 05 E6 85 09 A2 08 30
0330: A0 00 B1 1A 24 32 30 02 AA
0338: 49 7F A4 24 91 08 E6 1A 2E
0340: D0 02 E6 1B A5 09 18 69 A8
0348: 04 85 09 CA D0 E2 A5 45 22
0350: A6 46 A4 47 58 4C F0 FD 18
```

3-D Drawing Master

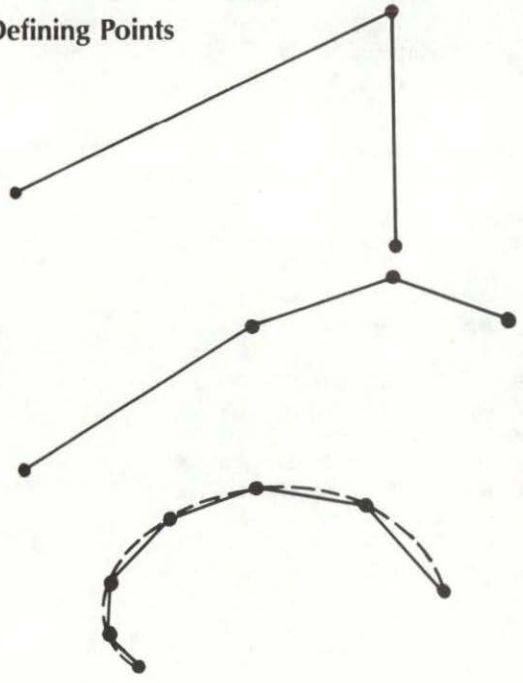
Donald E. Smith

"3-D Drawing Master" helps you create complex three-dimensional drawings by making use of your Apple computer's high-resolution graphics mode. For DOS 3.3 or ProDOS.

Drawing with a joystick is fun, but it lacks the precision and flexibility that artists and designers expect in a drawing instrument. With "3-D Drawing Master," you can create complex drawings on the screen. Just provide the points where each line begins and ends, and 3-D Drawing Master fills in the lines for you.

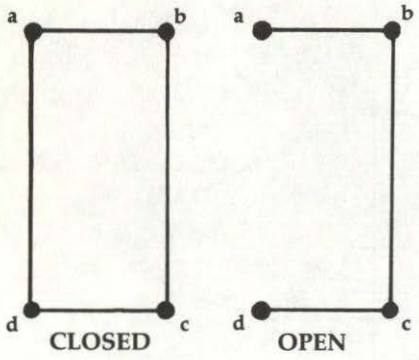
You can define the starting points of two lines and the point where they meet (Figure 1, top), or the places where a single line changes direction (Figure 1, middle), or you can define the points along a curved line (Figure 1, bottom).

Figure 1. Defining Points



You can also define open and closed two-dimensional shapes with 3-D Drawing Master.

Figure 2. Two Dimensions



Or you can even define complex drawings using a combination of two- and three-dimensional shapes.

Cartesian Coordinates

Drawing Master makes use of your computer's high-resolution graphics mode, which allows you to draw by controlling the individual pixels (specks of light) on the screen. In high-resolution mode, you find a specific pixel's position by using the Cartesian Coordinate System, the common X,Y coordinate system widely used in plotting charts and graphs. You can locate any point on the screen by its horizontal (X) and vertical (Y) distances from a point of origin. It's much like the way you locate a particular street on a city map by looking for it in the square called F–5 or C–2.

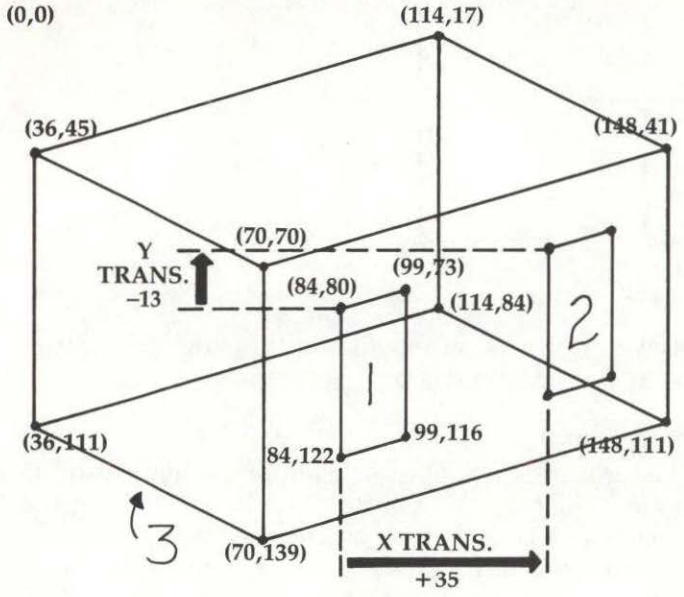
X-coordinate points begin at the left of the screen and increment to the right, and Y-coordinate points begin at the top of the screen and increment down. Position (0,0) is the HOME position, in the upper-left corner of the screen. So position (36,45) would be 36 X points to the right of and 45 Y points down from position (0,0).

The Apple II hi-res screen has a total of 53,760 individual points on the screen—280 X coordinates and 192 Y coordinates. You can use any of these points with 3-D Drawing Master.

Try It Out

To illustrate how Drawing Master uses this plotting system, let's create the drawing in Figure 3, step by step. (This plotting routine was taken from Paul F. Schatz's article, "High Resolution Plotting," in *COMPUTE!'s First Book of VIC.*)

Figure 3. Three-dimensional



First, type in 3-D Drawing Master and save it to disk. Load the program and run it. It begins by displaying a menu with seven options:

D—Define Object
R—Read Object
S—Save Object
V—View Object
T—Translate Object
M—Main Menu
E—End Session

Select D to define the object you wish to draw on the screen.

In this practice run, you'll be defining the different shapes contained in Figure 3, a box with two windows in the front. If you break down the drawing into its components, you have

one three-dimensional part (the box), and two two-dimensional parts (the windows).

Pressing D begins a series of prompts asking you to provide data for the object. The first prompt asks for the number of 2-D (two-dimensional) parts, which you can think of as the number of flat surfaces which you want to represent on the screen. In Figure 3, the box is drawn to simulate three dimensions, but the two windows are drawn as flat rectangles, so you should respond with 2 for the number of 2-D parts.

These two-dimensional parts are defined with the next series of prompts. You're asked for the number of points in part 1. Window 1 has four points, so enter 4. The next prompt which appears on the screen asks if you want the figure(s) open or closed (like Figure 2). Since the windows are closed rectangles, respond with Y.

Then, 3-D Drawing Master asks for the X and Y coordinates of each point. (Remember the X,Y coordinate system.) For the sake of clarity and consistency, enter the coordinates in clockwise order. Be sure you've entered each set of coordinates correctly before you press Return; 3-D Drawing Master has no editor, and a mistake can cause you to have a lopsided figure. Each time you press Return, you'll be prompted for the coordinates of the next point.

X,Y 84,80
X,Y 99,73
X,Y 99,116
X,Y 84,122

Since you have two windows (two 2-D parts), the next prompt asks for the data for the second window; 3-D Drawing Master allows you to make a second window from data already entered for the first. (If you wanted a third window, you would copy it from the completed second window.)

COPY LAST PART? (Y/N) *Y*

Answer with Y, because you want both windows to be alike, but in different positions on the screen. The next prompt asks for the translation of the figure in both X and Y directions. This determines how far horizontally (X) or vertically (Y) from the original position you want to copy the first window to produce the second window.

Take a look at Figure 3 again. In relation to the HOME position (0,0), window 2 is farther to the right and higher than

window 1. To make the example as painless as possible, the exact distances have been computed for you: Window 2 is to be translated 35 X points to the right of and 13 Y positions higher than window 1. Thus, 3-D Drawing Master displays

PART2/TRANSLATION:
RIGHT= +Y LEFT=−X
DOWN= +Y UP=−Y

This tells you that your X translation is positive if you're translating (moving) the figure to the right, and negative if translating to the left. The Y translation is positive if you're translating down, and negative if you're translating up from the original position.

Type 35 when you see X TRANS. Type −13 in answer to Y TRANS. Since the windows are not three-dimensional, you respond with N to the prompt

IS PART 3D? ***N***

You have yet to create the box which surrounds the windows.

When the prompt asks

 # OF 3D PARTS?

respond with 1, for the one box.

You can now create the 3-D simulation by creating certain sides of the box and letting 3-D Drawing Master fill in the rest for you. Look at Figure 4. This is a simulated three-dimensional drawing of a box similar to the one you're about to create. (The dotted lines would be hidden if the box were solid.) This box has six sides, or *faces*: front (face 1), back (face 2), top (face 3), bottom (face 4), and the left and right sides, which, to avoid clutter, are not labeled. Each face is a closed, four-sided shape.

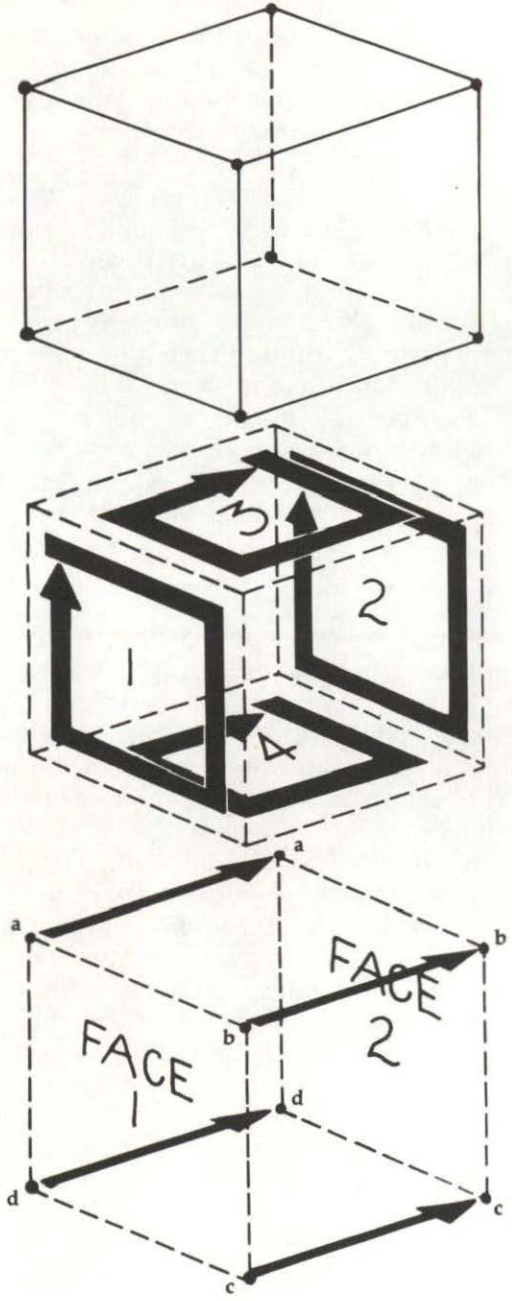
First, you're asked for the number of points on each face:

 # OF POINTS ON EACH FACE? ***4***
IS THE SIDE CLOSED? ***Y***

Now, enter the points for two opposite sides:

Face 1	**Face 2**
X,Y 36,45	**X,Y 114,17**
X,Y 70,70	**X,Y 148,41**
X,Y 70,139	**X,Y 148,111**
X,Y 36,111	**X,Y 114,84**

Figure 4. Faces



When you press Return after entering the fourth set of coordinates for face 2, you'll return to the main menu. From this menu, you can view your figure by pressing V. After you view it, press M to return to the main menu. You can save your drawings on disk (with the S option on the main menu) and load them back into memory by using the R option.

Translation

An extra utility in 3-D Drawing Master is the ability to move the entire object around the screen. Translation works like Copy. It prompts you for the X and Y translation numbers. It can be used for centering the object or for other purposes. It physically alters the database. If you don't wish to keep the translation after viewing it, don't save it.

The program makes it possible to view an object positioned half on and half off the screen. There's a safety device in 3-D Drawing Master that doesn't allow POKEing locations outside the screen area. The plotter will continue to compute those points, but they'll not be POKEd.

Viewing Tips

You may have noticed that after you pressed V, there was a pause and perhaps you saw some random graphics as the screen locations were being cleared out. This is normal. Even when there is no garbage displayed, the "clearing" process is going on. Be patient, and soon your image will begin to form. When you pressed M to return to the menu, the image was "cleared" in the same way.

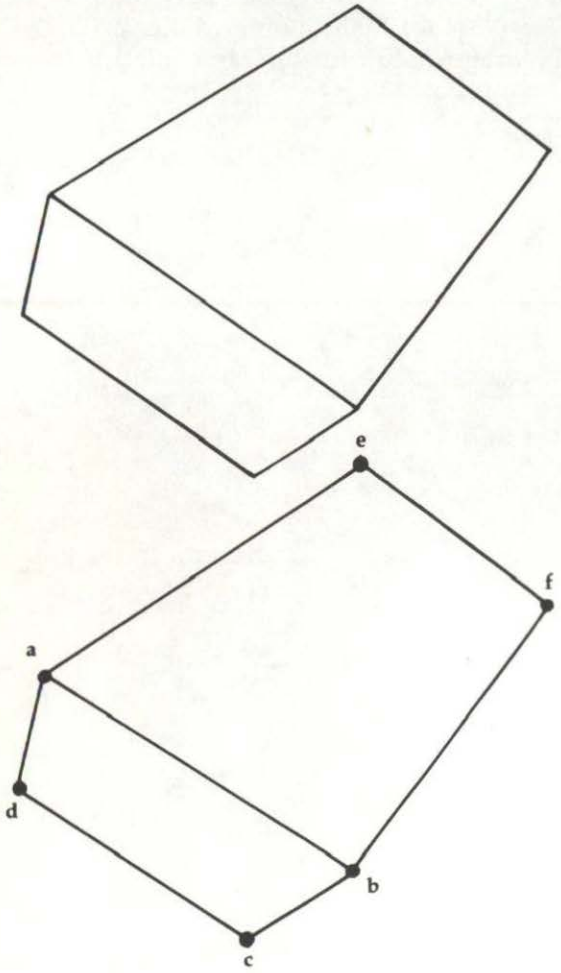
You may be puzzled by the way 3-D Drawing Master draws diagonal lines. Rather than drawing a straight line, it draws a very small staircase between two points. This is a product of the Apple's resolution. Any raster scan system, no matter how sharp the resolution, will give the same effect to some extent. The stairsteps are less noticeable if you draw as large as possible. Try to use the whole screen.

Hidden Lines in 3-D

Normally, 3-D mode draws a representation of the object as if it were transparent; you can see all edges at once. This is sometimes confusing. (If you're not careful, a simple cube may seem to rapidly change positions while you look at it. One moment it may seem like you're looking at it from the bottom,

the next from the top.) The 3-D Drawing Master program has no algorithms for removing hidden lines automatically, but under certain conditions you can fool the 3-D mode into drawing as if it did. Figure 5 (top) shows a hidden line drawing for a rectangular solid in perspective. Only the visible edges are showing. You could draw it in 2-D as two parts (two rectangular shapes). Or you can make 3-D draw it as one part. Here's the secret. Nothing prohibits you from using the same point *twice*.

Figure 5. Twice on the Same Point

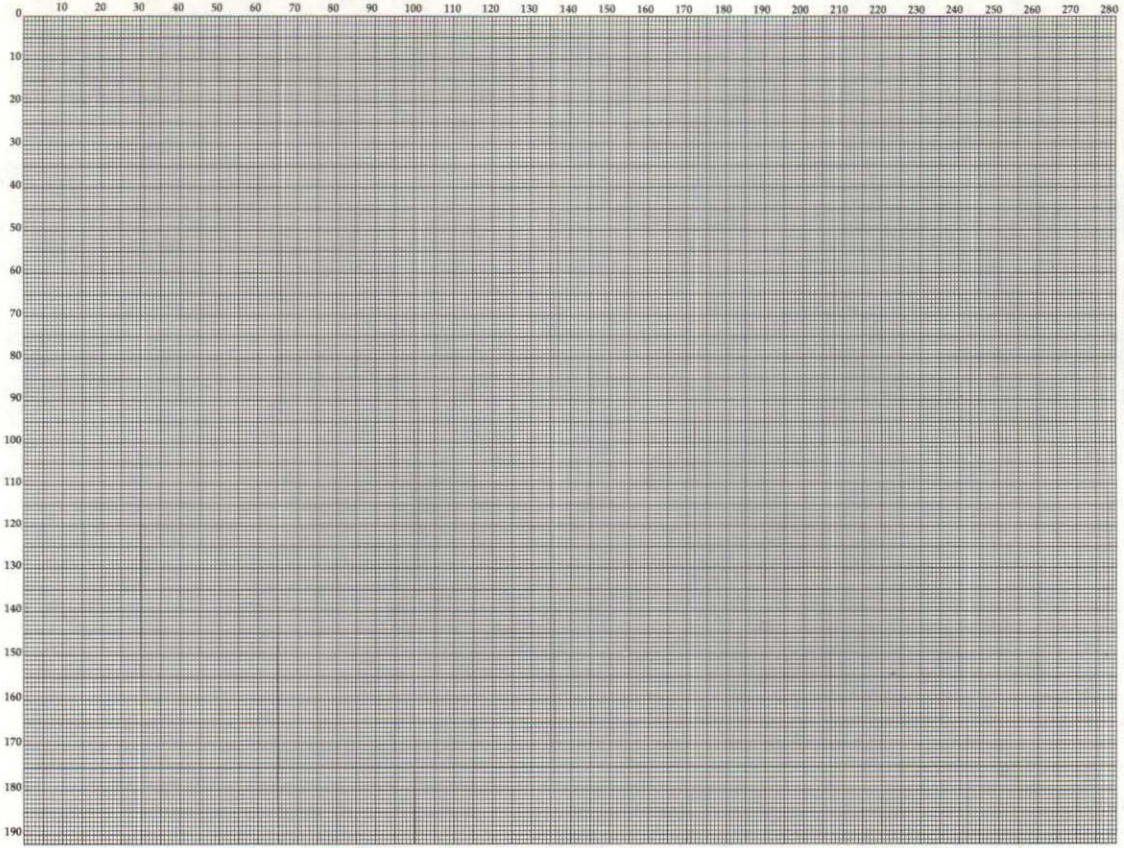


The first face is entered as usual, starting with point a, then b, c, and d. The second face is entered differently: Points e, f, b, and a. If your eyes are quick and you follow the way the Apple plots, you'll see that it traces the same line more than once. But the result is the image you want (Figure 5, bottom).

A Plotting Table

In order to define your object accurately in terms of its points, you'll need a plotting table like the one shown below in Figure 6. The object or scene you want to reproduce on the screen should be drawn or traced onto tissue paper and laid over the plotting table. Note on the drawing the X and Y coordinates for each important point. In standard notation, the X coordinate is given first, followed by the Y, with a comma separating them.

Figure 6. Plotting Table



3-D Drawing Master

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
E5 1020 DIM A(20,2,26)
7C 1030 Q$ = ""
66 1040 REM MENU
4C 1050 HOME
E9 1060 PRINT SPC( 13)"DRAWING MASTER": PRINT : PRIN
      T : PRINT
90 1070 PRINT SPC( 12)"D-DEFINE OBJECT": PRINT
AE 1080 PRINT SPC( 12)"R-READ OBJECT": PRINT
5F 1090 PRINT SPC( 12)"S-SAVE OBJECT": PRINT
3E 1100 PRINT SPC( 12)"V-VIEW OBJECT": PRINT
10 1110 PRINT SPC( 12)"T-TRANSLATE OBJECT": PRINT
C1 1120 PRINT SPC( 12)"M-MAIN MENU": PRINT
9D 1130 PRINT SPC( 12)"E-END SESSION"
5D 1140 GOSUB 3700
6D 1150 GP = (Q$ = "D") + 2 * (Q$ = "R") + 3 * (Q$ =
      "S") + 4 * (Q$ = "V") + 5 * (Q$ = "T")
06 1155 ON GP GOTO 1190,3000,3250,1670,2700
9B 1160 IF Q$ = "E" THEN 3740
7C 1170 GOTO 1050
86 1180 REM DEFINE
5E 1190 HOME
34 1200 REM 2D MODE
50 1210 PRINT "MAXIMUMS:": PRINT "20 PARTS": PRINT "
     12 PTS PER PART/FACE": PRINT
33 1220 INPUT "# OF 2D PARTS ? ";P1
C8 1230 PA = P1 + 2:Q$ = "":A(0,0,0) = P1
BA 1240 IF P1 = 0 THEN 1410
C3 1250 FOR PT = 2 TO P1 + 1
71 1260 PRINT : PRINT "PART ";PT - 1
DC 1270 IF PT = 2 THEN 1320
A2 1280 PRINT "COPY LAST PART? (Y/N) ": GOSUB 3700
BB 1290 IF Q$ = "N" THEN 1320
5B 1300 GOSUB 2430
7E 1310 GOTO 1390
09 1320 INPUT "# OF POINTS ? ";PO
A6 1330 PRINT "CLOSED ? (Y/N)": GOSUB 3700
1D 1340 IF Q$ = "Y" THEN A(PT,0,1) = 1: GOTO 1360
92 1350 A(PT,0,1) = 2
FD 1360 A(PT,0,0) = PO
CB 1370 TA = 2
86 1380 FOR T = 2 TO PO + 1: INPUT "X,Y ? ";A(PT,0,T
     A),A(PT,0,TA + 1):TA = TA + 2: NEXT
46 1390 NEXT PT
B8 1400 REM 3D MODE
54 1410 INPUT "# OF 3D PARTS ? ";P2
F7 1420 IF P2 = 0 THEN A(1,0,0) = 0: GOTO 1040
26 1430 Q$ = "":A(1,0,0) = P2
```

```
4A 1440 FOR PT = PA TO P2 + PA - 1
71 1450 PRINT : PRINT "PART";PT - 1
11 1460 IF (PT = PA) THEN 1510
A2 1470 PRINT "COPY LAST PART ? (Y/N)": GOSUB 3700
BB 1480 IF Q$ = "N" THEN 1510
81 1490 GOSUB 2430
6C 1500 GOTO 1630
21 1510 PRINT "# OF POINTS?"
BB 1520 INPUT "ON EACH FACE ? ";PO
F5 1530 A(PT,0,0) = PO
AE 1540 PRINT "CLOSED ? (Y/N) ": GOSUB 3700
2D 1550 IF Q$ = "Y" THEN A(PT,0,1) = 1: GOTO 1570
9A 1560 A(PT,0,1) = 2
CF 1570 TA = 2
1C 1580 FOR TT = 1 TO 2
08 1590 PRINT "FACE ";TT
28 1600 FOR T = 2 TO PO + 1: INPUT "X,Y ? ";A(PT,TT,
      TA),A(PT,TT,TA + 1):TA = TA + 2: NEXT
B9 1610 TA = 2
34 1620 NEXT TT
34 1630 NEXT PT
7A 1640 GOTO 1050
F6 1650 HGR2 : HCOLOR= 3
FE 1670 HGR2 : HCOLOR= 3
08 1740 REM 2D READ
0B 1750 P1 = A(0,0,0):PA = P1 + 2
4E 1760 IF P1 = 0 THEN 1900
D5 1770 FOR PT = 2 TO P1 + 1
85 1780 PO = A(PT,0,0)
DB 1790 TA = 2
B0 1800 FOR T = 2 TO PO + 1
41 1810 X1 = A(PT,0,TA):Y1 = A(PT,0,TA + 1)
94 1820 REM VIEW 1ST COMPARE
6E 1830 IF T = PO + 1 AND A(PT,0,1) = 1 THEN X2 = A(
      PT,0,2):Y2 = A(PT,0,3): GOSUB 2160
80 1840 IF T = PO + 1 THEN 1890
B2 1850 X2 = A(PT,0,TA + 2):Y2 = A(PT,0,TA + 3)
89 1860 GOSUB 2160
CA 1870 TA = TA + 2
A6 1880 NEXT T
50 1890 NEXT PT
7C 1900 REM 3D READ
B4 1910 P2 = A(1,0,0)
04 1920 IF P2 = 0 THEN 2340
66 1930 FOR PT = PA TO PPA - 1
B7 1940 PA = A(PT,0,0)
18 1950 FOR TT = 1 TO 2
D3 1960 TA = 2
CE 1970 FOR T = 2 TO PO + 1
0B 1980 X1 = A(PT,TT,TA):Y1 = A(PT,TT,TA + 1)
```

```
85 1990 IF T = PO + 1 AND A(PT,0,1) = 1 THEN X2 = A(
      PT,TT,2):Y2 = A(PT,TT,3): GOSUB 2160
42 2000 IF T = PO + 1 THEN 2050
E5 2010 X2 = A(PT,TT,TA + 2):Y2 = A(PT,TT,TA + 3)
6A 2020 GOSUB 2160
AB 2030 TA = TA + 2
B7 2040 NEXT T
35 2050 NEXT TT
C2 2060 TA = 2
AC 2070 FOR FA = 1 TO PO
62 2080 X1 = A(PT,1,TA):Y1 = A(PT,1,TA + 1)
A2 2090 X2 = A(PT,2,TA):Y2 = A(PT,2,TA + 1)
64 2100 GOSUB 2160
A5 2110 TA = TA + 2
F6 2120 NEXT FA
2B 2130 NEXT PT
74 2140 GOTO 2340
49 2150 REM VIEW 2ND COMPARE
8F 2160 GOSUB 3500: RETURN
8B 2330 REM VIEW GO MENU
62 2340 GOSUB 3700
85 2360 IF Q$ < > CHR$ (77) THEN 2340
29 2380 TEXT
63 2390 HOME
67 2400 GOTO 1050
CF 2410 END
B2 2420 REM COPY PART
A3 2430 HOME : PRINT "PART";PT - 1;"/TRANSLATION:"
F7 2440 PRINT "RIGHT=+X LEFT=-X
36 2450 PRINT "DOWN=+Y UP=-Y": PRINT
50 2460 INPUT "X TRANS.";TX
68 2470 INPUT "Y TRANS.";TY
BB 2480 PRINT "IS PART 3D?(Y/N)": GOSUB 3700
0E 2490 IF Q$ = "Y" THEN 2590
76 2500 A(PT,0,0) = A(PT - 1,0,0):A(PT,0,1) = A(PT -
      1,0,1)
B8 2510 TA = 2
7B 2520 FOR T = 2 TO A(PT,0,0) + 1
4E 2530 A(PT,0,TA) = A(PT - 1,0,TA) + TX
D5 2540 A(PT,0,TA + 1) = A(PT - 1,0,TA + 1) + TY
BD 2550 TA = TA + 2
99 2560 NEXT T
5F 2570 HOME
FC 2580 RETURN
9A 2590 A(PT,0,0) = A(PT - 1,0,0):A(PT,0,1) = A(PT -
      1,0,1)
FE 2600 FOR TT = 1 TO 2
BA 2610 TA = 2
7D 2620 FOR T = 2 TO A(PT,0,0) + 1
07 2630 A(PT,TT,TA) = A(PT - 1,TT,TA) + TX
```

```
79 2640 A(PT,TT,TA + 1) = A(PT - 1,TT,TA + 1) + TY
BF 2650 TA = TA + 2
9B 2660 NEXT T
49 2670 NEXT TT
65 2680 HOME
03 2690 RETURN
5E 2700 REM TRANS
98 2710 HOME : PRINT "TRANSLATION:"
F5 2720 PRINT "RIGHT=+X LEFT=-X
34 2730 PRINT "DOWN=+Y UP=-Y": PRINT
36 2740 INPUT "X TRANS. ? ";TX
5E 2750 INPUT "Y TRANS. ? ";TY
10 2760 P1 = A(0,0,0):PA = P1 + 2
D6 2770 FOR PT = 2 TO P1 + 1
86 2780 PO = A(PT,0,0)
DC 2790 TA = 2
B1 2800 FOR T = 2 TO PO + 1
1E 2810 A(PT,0,TA) = A(PT,0,TA) + TX
15 2820 A(PT,0,TA + 1) = A(PT,0,TA + 1) + TY
BB 2830 TA = TA + 2
97 2840 NEXT T
41 2850 NEXT PT
C7 2860 P2 = A(1,0,0)
C7 2870 FOR PT = P TO PP2 + PA - 1
88 2880 PO = A(PT,0,0)
27 2890 FOR TT = 1 TO 2
BC 2900 TA = 2
B7 2910 FOR T = 2 TO PO + 1
D6 2920 A(PT,TT,TA) = A(PT,TT,TA) + TX
9A 2930 A(PT,TT,TA + 1) = A(PT,TT,TA + 1) + TY
C1 2940 TA = TA + 2
9D 2950 NEXT T
4B 2960 NEXT TT
4B 2970 NEXT PT
91 2980 GOTO 1050
B1 2990 REM READ
4A 3000 HOME : INPUT "NAME OF PICTURE FILE? ";F$
21 3008 OU = 8
DB 3020 PRINT CHR$ (4);"OPEN";F$: PRINT CHR$ (4);"RE
      AD";F$
6A 3030 INPUT A(0,0,0),A(1,0,0)
2D 3040 FOR PT = 2 TO A(0,0,0) + 1
BF 3050 TA = 2
05 3060 INPUT A(PT,0,0),A(PT,0,1)
84 3070 FOR T = 1 TO A(PT,0,0) + 1
68 3080 INPUT A(PT,0,TA),A(PT,0,TA + 1)
C4 3090 TA = TA + 2
7A 3100 NEXT T
24 3110 NEXT PT
53 3120 FOR PT = A(0,0,0) + 2 TO A(1,0,0) + A(0,0,0)
        + 1
```

```
FA 3130  INPUT A(PT,0,0),A(PT,0,1)
06 3140  FOR TT = 1 TO 2
C1 3150  TA = 2
84 3160  FOR T = 2 TO A(PT,0,0) + 1
2E 3170  INPUT A(PT,TT,TA),A(PT,TT,TA + 1)
C2 3180  TA = TA + 2
9E 3190  NEXT T
26 3200  NEXT TT
26 3210  NEXT PT
AE 3220  PRINT CHR$ (4);"CLOSE";F$
70 3230  GOTO 1050
A8 3240  REM FILE
52 3250  HOME
66 3260  HOME : INPUT "NAME OF PICTURE FILE? ";F$
FB 3268  IN = 8
77 3270  PRINT CHR$ (4);"OPEN";F$: PRINT CHR$ (4);"WR
         ITE";F$
F5 3280  R$ = ","
C0 3290  PRINT A(0,0,0): PRINT A(1,0,0)
23 3300  FOR PT = 2 TO A(0,0,0) + 1
B4 3310  TA = 0
74 3320  FOR T = 0 TO A(PT,0,0) + 1
C3 3330  PRINT A(PT,0,TA): PRINT A(PT,0,TA + 1)
B6 3340  TA = TA + 2
92 3350  NEXT T
3C 3360  NEXT PT
6B 3370  FOR PT = A(0,0,0) + 2 TO A(1,0,0) + A(0,0,0)
         + 1
31 3380  PRINT A(PT,0,0): PRINT A(PT,0,1)
1E 3390  FOR TT = 1 TO 2
B3 3400  TA = 2
76 3410  FOR T = 2 TO A(PT,0,0) + 1
A9 3420  PRINT A(PT,TT,TA): PRINT A(PT,TT,TA + 1)
B4 3430  TA = TA + 2
90 3440  NEXT T
3E 3450  NEXT TT
3E 3460  NEXT PT
C6 3470  PRINT CHR$ (4);"CLOSE";F$
88 3480  GOTO 1050
5F 3490  REM PLOT SUB
68 3500  IF X1 > 279 OR X1 < 0 OR X2 < 0 OR X2 > 279
         THEN RETURN
9A 3510  IF Y1 > 190 OR Y1 < 0 OR Y2 < 0 OR Y2 > 190
         THEN RETURN
15 3515  HPLOT X1,Y1 TO X2,Y2: RETURN
74 3690  REM Q$
F5 3700  IF PEEK ( - 16384) < 127 THEN 3700
67 3710  GET Q$
E9 3720  RETURN
6A 3730  REM END SESSION
FF 3740  HOME : END
```

Apple Fractals

Paul W. Carlson

Fractals are receiving a lot of attention in mathematics and computer graphics these days. They're being used for everything from simulating random plant growth to generating realistic planetary landscapes for science-fiction films. This article introduces the fascinating world of fractals with three programs which demonstrate a particular type of fractal that can be plotted on an Apple II computer. For either DOS 3.3 or ProDOS.

The word *fractal* was coined by Benoit Mandelbrot, a pioneer in their study, to denote curves or surfaces having *fractional dimension*. The concept of fractional dimension can best be illustrated by considering a straight curve, or line. It's one-dimensional, having only length. But if the curve is infinitely long and curves about in such a manner as to completely fill an area of the plane containing it, the curve could be considered two-dimensional. A curve partially filling an area would have a fractional dimension between one and two.

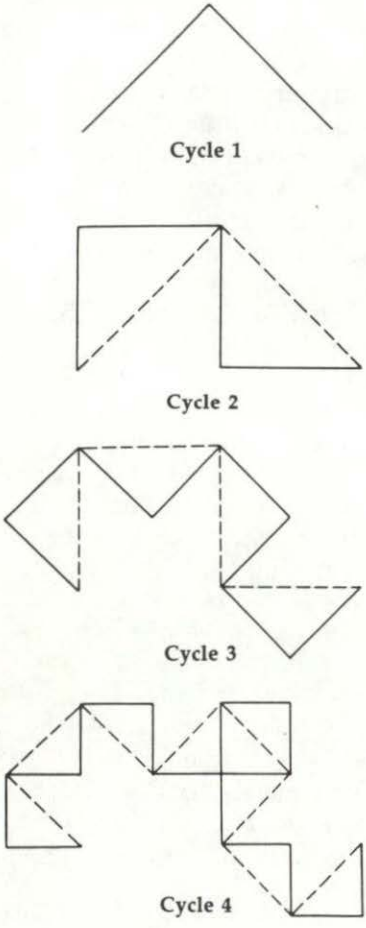
Many types of fractals are *self-similar*, which means that all portions of the fractal resemble each other. Self-similarity occurs whenever the whole is an expansion of some basic building block. In the language of fractals, this basic building block is called the *generator*. The generator in the accompanying programs consists of a number of connected line segments. The curves that the programs plot are the result of starting with the generator and then repeatedly replacing each line segment with the whole generator according to a defined rule. Theoretically, these replacement cycles would continue indefinitely. In practice, the screen resolution limits the number of cycles.

The programs illustrate two types of fractal curves. The curves generated by Program 1 and Program 2 are *self-contacting*, while the curve generated by Program 3 is *self-avoiding*. A self-contacting curve touches itself but does not cross itself. A self-avoiding curve never actually touches itself although it may appear to because of the limited screen resolution.

The Dragon Sweep

Program 1 plots what Mandelbrot refers to as a *dragon sweep.* It demonstrates, in a step-by-step fashion, how a fractal curve is filled. The generator consists of two-line segments of equal length forming a right angle. During each replacement cycle, the generator is substituted for each segment on alternating sides of the segments, that is, to the left of the first segment, to the right of the second segment, and so on. The program is written in BASIC, so the plotting is slow enough to let you see the development of the curve. Figure 1 shows the first few cycles of substitution.

Figure 1. Substitution



The program prompts you to enter an even number of cycles (for reasons of efficiency and screen resolution, only even numbers of cycles are plotted). When a plot is complete, pressing any key clears the screen and returns you to the prompt. I recommend starting with 2 cycles, then 4, 6, and so on. It takes 14 cycles to completely fill in the "dragon," but since this requires almost two hours, you'll probably want to quit after about 10 cycles. You can see the complete dragon by running Program 2, which always plots the dragon first in less than 30 seconds.

Since it's not at all obvious how the program works, here's a brief explanation. NC is the number of cycles; C is the cycle number; SN is an array of segment numbers indexed by cycle number; L is the segment length; D is the segment direction, numbered clockwise from the positive X direction; and X and Y are the high-resolution screen coordinates.

Line(s)	Function
100–140	Get number of cycles from user
150	Computes segment length
160	Sets starting coordinates
170	Sets segment numbers for all cycles to the first segment
180–220	Find the direction of the segment in the last cycle by rotating the segment in each cycle that will contain the segment in the last cycle
230–260	Increase or decrease X or Y by the segment length, depending on the segment direction
270–290	Plot the segment and update the current segment number for each cycle
300–320	If the segment number for cycle 0 is still zero, do the next segment; otherwise, the program's done.

Eight Thousand Dragons

Program 2 plots more than 8000 different dragons. It does this by randomly determining on which side of the first segment the generator will be substituted for all cycles after the first. The generator is always substituted to the left of the first segment in the first cycle to avoid plotting off the screen. Other than the randomization, this program uses the same logic as Program 1. The main part of this program is written in machine language to reduce the time required to plot a completely filled-in dragon from about two hours to less than half a minute.

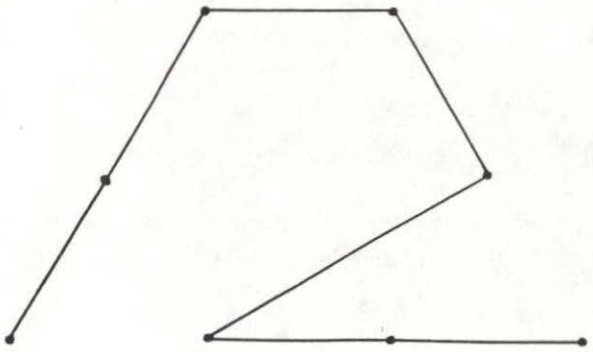
All the dragons are plotted after 14 cycles of substitution. All have exactly the same area, which equals half of the square of the distance between the first and last points plotted. All the dragons begin and end at the same points.

When a plot is complete, press the space bar to plot another dragon, or press the Q key to quit.

Snowflakes

Program 3 plots what Mandelbrot refers to as a *snowflake sweep*. The generator, shown in Figure 2, was discovered by Mandelbrot. The segments are numbered zero through six, starting at the right.

Figure 2. Snowflake Generator



The program is basically the same as Program 1. The variables NC, C, SN, D, X, and Y represent the same values, except that the direction D is numbered counterclockwise from the negative X direction. For each segment, the following table gives the value of RD (relative direction), LN (length factor), and SD (flags indicating which side of the segment the generator is to be placed).

Program 3 Values

Segment Number SN	Relative Direction RD	Length Factor LN	Side Flag SD
0	0	1/3	0
1	0	1/3	1
2	7	$\sqrt{1/3}$	1
3	10	1/3	0
4	0	1/3	0
5	2	1/3	0
6	2	1/3	1

Here's how Program 3 is organized:

Line(s)	Function
60	Reads values of SD and RD; computes LN values
70–90	Compute delta x and delta y factors for each direction
100–140	Get number of cycles from user
160	Sets starting coordinates
170	Sets the segment numbers for all cycles to the first segment
180–210	Find the direction of the segment in the last cycle
220–230	Compute the coordinates of the end of the segment, plot the segment, and update the segment numbers for each cycle
240–260	Same as lines 300–320 in Program 1

Like Program 1, pressing any key when a plot is complete clears the screen and brings another prompt.

Experiment!

I hope these programs encourage you to look further into the fascinating world of fractals. Don't be afraid to experiment with the programs—try modifying the shape of the generator in Program 3, for example. Better yet, design your own generator.

These programs just begin to explore the possibilities of fractal computer graphics. There is another whole class of fractals, those generated by functions of complex variables. And then there are three-dimensional fractals. And then....

Program 1. The Dragon Sweep

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in the following three programs.

```
1E 10 REM  PROGRAM 1
6A 20 REM
7B 30 REM  THIS PROGRAM PLOTS A FRACTAL "DRAGON SWEE
      P"
D0 40 REM  FOR AN EVEN NUMBER OF CYCLES (14 MAX).
6D 50 REM
9D 90 DIM SN(14)
54 100 TEXT : HOME
F1 110 PRINT "ENTER AN EVEN NO. OF CYCLES (2 TO 14)"
98 120 INPUT "           OR ENTER A ZERO TO QUIT: ";N
      C
A7 130 IF NC = 0 THEN END
E4 140 IF INT (NC / 2) # 2 < > NC OR NC < 2 OR NC >
      14 THEN 100
1D 150 L = 128: FOR C = 2 TO NC STEP 2:L = L / 2: NE
      XT
E8 160 X = 77:Y = 128: HGR2 : HCOLOR= 3: HPLOT X,Y
81 170 FOR C = 0 TO NC:SN(C) = 0: NEXT
43 180 D = 0: FOR C = 1 TO NC: IF SN(C - 1) = SN(C)
      THEN D = D - 1: GOTO 200
46 190 D = D + 1
ED 200 IF D = - 1 THEN D = 7
1C 210 IF D = 8 THEN D = 0
FD 220 NEXT
9D 230 IF D = 0 THEN X = X + L: GOTO 270
F0 240 IF D = 2 THEN Y = Y + L: GOTO 270
A4 250 IF D = 4 THEN X = X - L: GOTO 270
9A 260 Y = Y - L
35 270 HPLOT TO X,Y:SN(NC) = SN(NC) + 1
10 280 FOR C = NC TO 1 STEP - 1: IF SN(C) < > 2 THEN
       300
9F 290 SN(C) = 0:SN(C - 1) = SN(C - 1) + 1: NEXT
BA 300 IF SN(0) = 0 THEN 180
D6 310 GET A$: IF A$ = "" THEN 310
98 320 GOTO 100
```

Program 2. Eight Thousand Dragons

```
2E 10 REM  PROGRAM 2
6A 20 REM
6B 30 REM
92 40 REM  THIS PROGRAM PLOTS RANDOM FRACTAL "DRAGON
       SWEEPS."
7C 50 REM  THE "STANDARD" DRAGON IS ALWAYS PLOTTED F
      IRST.
```

```
6E 60 REM
5F 70 REM  WHEN A PLOT IS COMPLETE, PRESS THE SPACE
     BAR
D1 80 REM  TO PLOT ANOTHER DRAGON, OR PRESS THE "Q"
     KEY
97 90 REM  TO EXIT THE PROGRAM.
82 100 REM
88 130 REM
6B 140 HIMEM: 16383
DB 150 FOR N = 24612 TO 24912: READ I: POKE N,I: NEX
     T
9F 160 FOR N = 24591 TO 24605: POKE N,0: NEXT : GOTO
      180
17 170 FOR N = 24593 TO 24605: POKE N, INT ( RND (1)
      * 2): NEXT
24 180 HGR2 : HCOLOR= 3: CALL 24619
85 190 GET A$: IF A$ = " " THEN 170
D8 200 IF A$ < > "Q" THEN 190
FF 210 TEXT : END
F0 220 DATA 1,2,4,8,16,32,64,169
64 230 DATA 0,141,16,96,160,14,153,0
1C 240 DATA 96,136,192,255,208,248,141,32
AF 250 DATA 96,162,77,142,31,96,160,128
22 260 DATA 140,33,96,32,248,96,169,0
A5 270 DATA 141,30,96,162,0,160,1,185
DB 280 DATA 15,96,208,20,238,30,96,189
26 290 DATA 0,96,217,0,96,208,26,206
2B 300 DATA 30,96,206,30,96,76,125,96
A8 310 DATA 206,30,96,189,0,96,217,0
26 320 DATA 96,208,6,238,30,96,238,30
85 330 DATA 96,173,30,96,16,5,169,7
AF 340 DATA 141,30,96,201,8,208,5,169
16 350 DATA 0,141,30,96,232,200,224,14
DB 360 DATA 208,189,170,208,20,173,31,96
07 370 DATA 24,105,1,141,31,96,173,32
4D 380 DATA 96,105,0,141,32,96,76,210
7A 390 DATA 96,224,2,208,6,238,33,96
44 400 DATA 76,210,96,224,4,208,20,173
0C 410 DATA 31,96,56,233,1,141,31,96
53 420 DATA 173,32,96,233,0,141,32,96
E1 430 DATA 76,210,96,206,33,96,32,248
15 440 DATA 96,238,14,96,160,14,162,13
6B 450 DATA 185,0,96,201,2,208,12,169
B4 460 DATA 0,153,0,96,254,0,96,202
CF 470 DATA 136,208,237,173,0,96,208,3
E1 480 DATA 76,74,96,96,173,33,96,10
D1 490 DATA 10,41,28,9,64,133,27,173
28 500 DATA 33,96,74,74,74,74,41,3
FF 510 DATA 5,27,133,27,173,33,96,41
45 520 DATA 192,72,106,133,26,104,74,74
```

```
1F 530 DATA 74,5,26,133,26,173,31,96
8F 540 DATA 141,34,96,173,32,96,141,35
66 550 DATA 96,56,160,255,200,173,34,96
0C 560 DATA 233,7,141,34,96,173,35,96
35 570 DATA 233,0,141,35,96,16,237,173
FC 580 DATA 34,96,105,7,170,189,36,96
71 590 DATA 17,26,145,26,96
```

Program 3. The Snowflake Sweep

```
3E 10 REM  PROGRAM 3
6A 20 REM
B0 30 REM  THIS PROGRAM PLOTS A FRACTAL "SNOWFLAKE S
     WEEP"
6C 40 REM
9C 50 DIM DX(11),DY(11):M = 7 / 6
1C 60 FOR N = 0 TO 6: READ SD(N),RD(N):LN(N) = 1 / 3
     : NEXT :LN(2) = SQR (LN(1))
F1 70 A = 0: FOR D = 6 TO 11:DX(D) = COS (A):DY(D) =
      SIN (A)
BC 80 A = A + 0.52359879: NEXT
EB 90 FOR D = 0 TO 5:DX(D) = - DX(D + 6):DY(D) = - D
     Y(D + 6): NEXT
54 100 TEXT : HOME
85 110 PRINT "ENTER NUMBER OF CYCLES ( 1 - 4 )"
90 120 INPUT "          OR ENTER A ZERO TO QUIT: ";NC
A7 130 IF NC = 0 THEN END
1A 140 IF NC > 4 THEN 100
9D 150 HGR2 : HCOLOR= 3
BE 160 X = 235:Y = 142:TL = 162: HPLOT X,Y
81 170 FOR C = 0 TO NC:SN(C) = 0: NEXT
84 180 D = 0:L = TL:NS = 0: FOR C = 1 TO NC:I = SN(C
     ):L = L * LN(I):J = SN(C - 1):NS = NS + SD(J)
     :K = INT (NS / 2): IF K * 2 < > NS THEN D = D
      + 12 - RD(I): GOTO 200
61 190 D = D + RD(I)
92 200 IF D > 11 THEN D = D - 12
FB 210 NEXT
70 220 X = X + M * L * DX(D):Y = Y - L * DY(D): HPLO
     T TO X,Y:SN(NC) = SN(NC) + 1: FOR C = NC TO 1
      STEP - 1: IF SN(C) < > 7 THEN 240
93 230 SN(C) = 0:SN(C - 1) = SN(C - 1) + 1: NEXT
C1 240 IF SN(0) = 0 THEN 180
4E 250 GET A$: IF A$ = "" THEN 250
97 260 GOTO 100
41 270 DATA 0,0,1,0,1,7,0,10,0,0,0,0,2,1,2
```

Automatic Scaling Plotter

R. R. Hiatt

This program scales and plots a set of points on any Apple II–series computer screen, eliminating any paper and pencil figuring. For DOS 3.3 or ProDOS.

Deciding on the best scaling system and actually plotting a multitude of points by hand is tedious to say the least. With "Automatic Scaling Plotter," the Apple can do it all for you. You just input the data.

Automatic Scaling Plotter uses its own character set for numbering the axes and for plotting the points. Actually, any standard set will do, though the one included here is slightly unusual, being based on a 5 × 7 dot matrix with the DRAW position at the top rather than being space-centered. Larger, space-centered characters would require some alteration in the integer offsets used in the DRAW routine (lines 560–585 of the program). In fact, you could create your own character set using "Apple SuperFont" and with some modification to the Plotter program, use "HROUT" to display these characters (both Apple SuperFont and HROUT are included in this book).

Fortunately, you don't have to modify Plotter if you use "CHARSET," the character set created for it. However, you *do* have to type it in and make sure it's on the same disk as Plotter.

Type in and save Automatic Scaling Plotter, Program 1. Next, load "AppleMLX" into your computer (if you haven't already typed in AppleMLX, refer to Appendix C for the program and its instructions). Run AppleMLX and respond to the starting and ending address prompts with

STARTING ADDRESS? 4000
ENDING ADDRESS? 4327

Type in CHARSET, Program 2, using AppleMLX. Save it to the same disk as Plotter, making sure you specify CHARSET

as the program's name (otherwise, Plotter won't recognize it). You're now ready to use Automatic Scaling Plotter.

Entering Data

When you run Automatic Scaling Plotter, it asks for the number of points you want to plot. Type in the appropriate number. The maximum number of points Plotter can display is 98—more than that and you'll receive an error.

Plotter can also randomly place points as a demonstration. To see this work, just answer Y at the next prompt, *RND POINTS?*. To enter your own point data, type N at this prompt and proceed to type in the coordinates in the normal X,Y format. Make sure you separate the X and Y coordinates with a comma.

As soon as you've entered your own coordinate data (or asked the computer to randomly generate its own), the computer will pause a moment or two as it calculates. You'll be asked to provide a character to be used to plot the points. Since this is a custom character set, its values are not the normal ASCII numbers. The ones you'll most likely use, the period (.) and the X, are 14 and 56, respectively.

The screen clears and your (or the computer's) points are shown, as well as the scale used for the plot. A Y-axis regression line is also drawn on the screen. That's all there is to it.

If you want to run the program again, simply type RUN and press Return.

Some Plotter Facts

Given the data, XMAX, XMIN, YMAX, YMIN, and thus ▲X and ▲Y are found. A subroutine determines whether the spread will be accommodated best by segmenting each axis in four, five, or six units. Three sets of scale factors are computed: PX,PY (powers of 10 that reduce X and Y to small numbers); ZX,ZY (zero offsets); and SX,SY (scale to screen dimensions). The arrays PX( ),PY( ) and X$( ),Y$ ( ) hold the information for positioning and numbering the axes.

Another useful subroutine computes the regression line of Y on X.

While the program given here is for split-screen HGR, it's easily adaptable to full-screen or to any part of the screen. Line 30 defines parameters in terms of screen dimensions; AX (length of X-axis), AY (length of Y-axis), YM (max permitted

screen Y), and BX and BY (baseline offsets). All other parameters and scale factors are computed in terms of these fundamental ones. The only restrictions are obvious: YM <= 191 (full screen) and (AX + BX) <= 279.

Finally, in plotting multiple sets of closely related data, it is frequently not desirable to rescale for each set. An example is plotting pH profiles for acid-base titrations where 0 <= pH <= 14 and 0 <= mL added Base < = 20. Initial input of two points, X_1,Y_1 = 0,0 and X_2,Y_2 = 20,14, establishes the scaling factors and the information for drawing and numbering the axes. Subsequent data sets, computed by the algorithm, are simply scaled and plotted, with screen erase and axis redrawing as you desire.

Program 1. Automatic Scaling Plotter

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
51 10 REM SCALER ROUTINE FOR APPLEPLOT
5B 15 D$ = CHR$ (4)
D8 20 DIM X(100),Y(100),XP(10),YP(10),X$(10),Y$(10)
86 25 IF PEEK (233) < > 64 OR PEEK (16384) < > 63 TH
      EN PRINT D$;"BLOAD CHARSET": POKE 232,0: POKE
      233,64
C9 30 AX = 240:BX = 31:AY = 145:BY = 12:YM = 159
F0 35 REM INPUT DATA
E3 40 INPUT "# OF POINTS ? ";N
0F 45 INPUT "RND POINTS ? ";Q$: IF Q$ < "Y" THEN 50
B6 47 FOR I = 1 TO N:X(I) = 100 * RND (I) * RND (I):
      Y(I) = 10 * RND (I): NEXT : GOTO 70
18 50 FOR I = 1 TO N: INPUT X(I),Y(I): NEXT
BA 60 REM FIND XMAX,XMIN,YMAX,YMIN
08 70 XU = X(1):XL = X(1):YU = Y(1):YL = Y(1)
92 80 FOR I = 2 TO N
D2 90 IF X(I) < XL THEN XL = X(I)
A3 100 IF X(I) > XU THEN XU = X(I)
55 105 NEXT : GOSUB 1000: FOR I = 2 TO N
74 110 IF Y(I) < YL THEN YL = Y(I)
D4 120 IF Y(I) > YU THEN YU = Y(I)
FE 130 NEXT
B4 140 DX = ABS (XU - XL):DY = ABS (YU - YL)
14 150 PRINT : PRINT "CALCULATING"
8A 190 REM FIND SCALE FACTOR AND POSITIONS TO LABEL
      X-AXIS
E0 200 UL = XU:LL = XL:DE = DX:AZ = AX
07 210 GOSUB 2000:S1 = S
1E 220 XU = UL:XL = LL:DX = DE:PX = PT:ZX = ZE:SX =
      SP:XS = SF
```

```
0E 230 FOR I = 1 TO 7:PL = INT (I * SX * XS + .5): I
       F PL > AX THEN 250
96 240 XP(I) = PL + BX:X$(I) = STR$ (ZX + SX * I): I
       F LEN (X$(I)) > 4 THEN X$(I) = LEFT$ (X(I),4)
04 250 NEXT
B1 255 REM SCALE X
5D 260 FOR I = 1 TO N:X(I) = BX + INT ((X(I) * (10 ^
        PX) - ZX) * XS + .5): NEXT
7B 300 REM FIND SCALE FACTOR AND POSITIONS TO LABEL
       Y-AXIS
3E 310 UL = YU:LL = YL:DE = DY:AZ = AY
4A 320 GOSUB 2000:S2 = S
01 330 YU = UL:YL = LL:DY = DE:PY = PT:ZY = ZE:SY =
       SP:YS = SF
EB 340 FOR I = 1 TO 7:PL = INT (I * SY * YS + .5): I
       F PL > AY THEN 360
AC 350 YP(I) = PL
07 360 NEXT
F4 370 FOR I = 1 TO N:Y(I) = INT ((Y(I) * (10 ^ PY)
       - ZY) * YS + .5): NEXT
D6 380 FOR I = 1 TO S2 - 1:YP(I) = YM - BY - YP(I):Y
       $(I) = STR$ (ZY + SY * I): IF LEN (Y$(I)) > 4
        THEN Y$(I) = LEFT$ (Y$(I),4)
1F 385 NEXT
23 390 FOR I = 1 TO N:Y(I) = YM - BY - Y(I): NEXT
B9 400 PRINT : PRINT : INPUT "WHAT NUMBER FOR CHAR ?
        ";CH
9E 430 PRINT : PRINT : PRINT : PRINT
A2 490 REM DRAW AND LABEL AXES; PLOT POINTS
F6 500 HGR
38 505 SCALE= 1
98 510 HCOLOR= 5
F3 515 XX = YM - BY
5F 520 HPLOT BX,0 TO BX,XX TO AX + BX,XX
CB 530 FOR I = 1 TO S2 - 1: HPLOT BX,YP(I) TO BX + 5
       ,YP(I): NEXT
96 540 HCOLOR= 3
A8 550 FOR I = 1 TO S1 - 1: HPLOT XP(I),XX - 5 TO XP
       (I),XX: NEXT
8F 560 FOR I = 1 TO S2 - 1:L = LEN (Y$(I)):X = 22
2B 565 FOR J = L TO 1 STEP - 1: XDRAW ASC ( MID$ (Y$
       (I),J,1)) - 32 AT X,YP(I) - 3
E0 570 X = X - 7: NEXT : NEXT
55 575 Y = INT (YM - BY / 2): FOR I = 1 TO S1 - 1:L
       = LEN (X$(I)):X = XP(I) - 3.5 * (L - 1)
2F 580 FOR J = 1 TO L: XDRAW ASC ( MID$ (X$(I),J,1))
        - 32 AT X,Y:X = X + 7
E2 585 NEXT : NEXT
E7 590 IF CH > 0 THEN 635
FB 600 FOR I = 1 TO N - 2
```

```
EB 610 HPLOT X(I) - 2,Y(I) TO X(I) + 2,Y(I)
41 620 HPLOT X(I),Y(I) - 2 TO X(I),Y(I) + 2
C8 630 NEXT : GOTO 640
F5 635 FOR I = 1 TO N - 2: XDRAW CH AT X(I),Y(I): NE
       XT
7B 637 REM PLOT LINEAR REGRESSION LINE FOR Y ON X
17 640 HPLOT X(N - 1),Y(N - 1) TO X(N),Y(N)
E6 650 VTAB (23)
82 660 PRINT "X-SCALE = 10^"PX; SPC( 3);
BF 670 PRINT "Y-SCALE = 10^"PY
A0 690 END
5B 998 REM DETN END POINTS OF LINEAR REGRESSION LINE
        AND ADD TO X,Y ARRAYS
2F 1000 T1 = 0:T2 = 0:T3 = 0:T4 = 0:T5 = 0
3F 1010 FOR I = 1 TO N
D2 1020 T1 = T1 + X(I):T2 = T2 + X(I) * X(I)
53 1030 T3 = T3 + Y(I):T4 = T4 + Y(I) * Y(I)
4B 1040 T5 = T5 + X(I) * Y(I)
B5 1050 NEXT
AE 1060 B = (T5 - T1 * T3 / N) / (T2 - T1 * T1 / N)
CC 1070 A = T3 / N - B * T1 / N
C3 1080 X(N + 1) = XL:X(N + 2) = XU
04 1090 Y(N + 1) = A + B * XL:Y(N + 2) = A + B * XU
3E 1100 N = N + 2
D7 1110 RETURN
E0 1990 REM SEARCH FOR BEST SCALE
CB 2000 PT = 0
A7 2010 IF DE < 10 THEN DE = 10 * DE:UL = 10 * UL:LL
        = 10 * LL:PT = PT + 1: GOTO 2010
BF 2020 IF DE > 100 THEN DE = DE / 10:UL = UL / 10:L
       L = LL / 10:PT = PT - 1: GOTO 2020
E8 2030 SP = 1.25: FOR I = 1 TO 4:SP = 2 * SP:S = DE
        / SP: IF INT (S) < > S THEN S = INT (S + 1)
B9 2040 IF S = 4 OR S = 5 OR S = 6 THEN I = 4
B6 2050 NEXT
F6 2060 ZE = SP * INT (LL / SP):SF = AZ / (UL - ZE)
EE 2070 RETURN
```

Program 2. CHARSET

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program.

```
START ADDRESS: 4000
END ADDRESS:   4327

4000: 3F 00 80 00 86 00 8B 00 7B
4008: 9A 00 AA 00 B5 00 C2 00 5E
4010: C6 00 CE 00 D6 00 E3 00 4C
4018: EC 00 F1 00 F6 00 FB 00 FC
```

```
4020: 03 01 12 01 1D 01 2D 01 FC
4028: 38 01 46 01 54 01 62 01 4A
4030: 6A 01 77 01 85 01 89 01 69
4038: 8E 01 99 01 A1 01 A9 01 E8
4040: B5 01 C4 01 D3 01 DF 01 E7
4048: E9 01 F4 01 00 02 0A 02 CA
4050: 17 02 25 02 2D 02 36 02 81
4058: 47 02 4E 02 60 02 6F 02 D2
4060: 7A 02 83 02 90 02 9E 02 FA
4068: AB 02 B7 02 C1 02 CA 02 04
4070: D7 02 E5 02 EE 02 F8 02 AD
4078: 06 03 0C 03 18 03 1E 03 7A
4080: 09 36 36 16 06 00 31 6E 3C
4088: 24 06 00 31 36 36 6E 24 3B
4090: 24 24 D6 1F 16 0D 0D 04 FB
4098: 30 00 09 36 36 36 07 38 86
40A0: 04 58 60 0D B5 F2 04 F8 90
40A8: 06 00 35 6F 09 1E 17 1E D6
40B0: 17 4D 35 37 00 A9 B6 16 54
40B8: 3F 20 04 20 56 4D B2 07 B7
40C0: 30 00 09 36 06 00 09 1E 3E
40C8: 17 36 0E 15 06 00 09 15 CC
40D0: 0E 36 1E 17 06 00 72 2D 5D
40D8: 05 F8 B4 32 36 1F 60 0D CF
40E0: 15 06 00 89 36 36 07 58 F6
40E8: 38 4D 35 00 92 09 F6 06 2C
40F0: 00 92 2D 2D 06 00 92 52 36
40F8: 11 06 00 49 11 1E 17 1E 65
4100: 17 06 00 29 AD 36 36 1E F2
4108: 3F 07 20 24 6C 11 1E 17 2D
4110: 06 00 09 36 36 36 6F 04 87
4118: 58 C0 03 30 00 29 AD F6 51
4120: 3F 17 36 2D 2D DF 03 58 E8
4128: 58 58 58 30 00 2D 2D F6 01
4130: 3E 9F 72 2D 05 20 06 00 8F
4138: 49 36 36 36 07 58 38 27 48
4140: 0C 05 B0 89 06 00 2D 2D 70
4148: 96 36 1E 3F 07 20 58 20 E4
4150: 15 2D 06 00 09 2D 96 32 C5
4158: 1E 3F 07 20 24 0C 16 2D 47
4160: 06 00 2D 2D F6 17 1E 36 E4
4168: 06 00 29 AD B6 F6 3F 07 05
4170: 20 04 20 95 2D 06 00 29 0C
4178: AD 36 F6 17 3F 04 58 58 C2
4180: 20 95 2D 06 00 91 B1 06 2E
4188: 00 91 B1 F6 06 00 49 1E F5
4190: 17 16 0E 15 DF 58 58 30 78
4198: 00 12 2D 2D 16 3F 3F 06 4A
41A0: 00 A9 0E 15 1E 17 1E 06 30
41A8: 00 29 AD F6 37 16 1F 58 43
```

```
41B0: 58 58 58 30 00 29 AD 36 B9
41B8: 3F B4 B5 3F 07 20 24 B4 68
41C0: 89 4A 32 00 09 15 0E 36 CF
41C8: 36 07 58 38 3F 24 0C 96 EF
41D0: 1A 36 00 2D AD B6 F6 3F 36
41D8: 27 24 24 95 2D 06 00 29 80
41E0: AD 96 F2 3F 07 20 24 34 67
41E8: 00 2D AD 36 36 1E 3F 27 9F
41F0: 24 24 06 00 2D 2D 96 3B D5
41F8: 3F 24 96 36 2D 2D 06 00 84
4200: 2D 2D 96 3B 3F 24 96 36 DA
4208: 06 00 29 2D B6 32 3E 3F C1
4210: 07 20 24 AC 4A 32 00 36 C0
4218: 36 36 04 40 2D 35 36 04 78
4220: 58 58 20 34 00 29 F5 36 F4
4228: 36 3E 0D 06 00 49 31 36 17
4230: 36 1E 3F 07 30 00 36 36 D3
4238: 36 04 40 15 0E 15 04 58 57
4240: 58 58 58 F0 17 06 00 36 28
4248: 36 36 2D 2D 06 00 36 36 C0
4250: 36 44 21 64 35 36 36 04 5E
4258: 58 58 58 58 F8 13 06 00 CF
4260: 36 36 36 04 58 40 15 0E 90
4268: 35 26 58 20 24 06 00 29 80
4270: AD 36 36 1E 3F 07 20 24 7C
4278: 34 00 2D AD F6 3F 27 B4 4F
4280: 32 36 00 29 AD 36 B6 1F 11
4288: 3F 20 24 B4 89 0E 06 00 15
4290: 2D AD F6 3F 27 B4 32 66 C0
4298: 09 15 07 C0 06 00 29 AD 04
42A0: 96 F6 3F 07 20 58 20 95 BE
42A8: 2D 06 00 2D 2D FE 36 36 20
42B0: FE 58 58 58 58 30 00 36 15
42B8: 36 76 2D 05 20 24 24 06 CB
42C0: 00 36 0E 76 26 08 64 24 3A
42C8: 06 00 36 36 76 0D 05 20 8C
42D0: 24 24 96 1B 36 06 00 76 35
42D8: 0D 05 20 96 12 26 18 1F 0B
42E0: 17 26 48 30 00 76 0D 05 7F
42E8: 20 96 1B 36 36 00 2D 2D 23
42F0: F6 17 1E 17 2E 2D 35 00 7C
42F8: 2D 2D DE 3B 36 36 2E 2D 03
4300: FD 03 20 24 06 00 72 15 B6
4308: 0E 15 06 00 2D 2D 36 36 5C
4310: 36 3F 3F 4C 21 24 34 00 30
4318: 91 2A AD DF 33 00 92 92 F6
4320: 2D 2D 06 00 20 41 54 20 18
```

5

Utilities and Programming Aids

Apple IIc RAM Disk Mover

Christopher J. Flynn

In addition to demonstrating the RAM disk and subdirectory options with ProDOS and an Apple IIc, this article presents a utility for rapidly copying a number of programs from the floppy drive to the RAM drive. Only for Apple IIc computers with ProDOS.

One of the conveniences of the Apple IIc is its built-in disk drive. An Apple disk holds about 143,360 characters, or 140K bytes, of information. (One K is equal to 1024 bytes.)

Typically, people use a disk to store both programs and data. If a disk contains programs that total about 40K, then only 100K remains for data. So a single-drive system can be a bit limiting in terms of storage capabilities.

One answer is to buy a second disk drive. But wait—there's an alternative you should consider first. Did you know that your Apple IIc has a second built-in disk drive that will hold about 60K of information? Of course, it's not a regular mechanical disk drive. Rather, it's an electronic drive known as a *RAM disk*. A RAM disk may sound like some futuristic propulsion mechanism, but it's really just a section of random access memory that, with the proper software, works like a disk drive. If you could use the RAM disk for program storage, you could use the conventional disk drive entirely for data storage. This article will show you how.

How the RAM Disk Works

The Apple IIc has 128K of RAM organized as two separate 64K sections, or *banks*. Each bank is addressed from location 0 through location 65535. Most programs, including Applesoft BASIC, are designed to use only the first 64K bank. Thus, the second 64K bank is generally free for use as a RAM disk. Some programs, such as the new releases of Logo and Pascal, do, however, use the entire 128K.

The Apple IIc's disk operating system, ProDOS, has the

necessary software to use the second 64K bank as a RAM disk. ProDOS can make the second bank look like any other disk drive to the computer.

The big advantage of the RAM disk (besides the fact that it's free) is its speed. It's entirely electronic, so there are no moving parts to slow down data access.

However, this characteristic is also the RAM disk's biggest disadvantage. Because RAM chips require constant power to maintain their information, the RAM disk forgets everything the instant the power goes off. In other words, you still have to use the mechanical drive for permanent storage.

Accessing the RAM Disk

Here's a simple experiment you can do. It shows how RAM program files work.

- Insert the ProDOS *System Utilities* disk in your IIc. Turn on the computer. Exit the *Utilities* program. You should be in Applesoft and ProDOS should be active.
- Check to see what, if anything, is stored in the RAM disk. Type

  CAT /RAM

  There should be 120 blocks available. At 512 bytes per block, this gives 61,440 bytes of storage—about 43 percent of the capacity of a floppy disk. This should certainly hold a few programs.
- Try saving a program with the RAM disk. Enter the following one-line program:

```
10 PRINT "I AM IN THE RAM DISK"
```

  Save the program by typing

  SAVE /RAM/DEMO
- Erase the program from memory by typing NEW. Now load the program by entering

  LOAD /RAM/DEMO

  or

  RUN /RAM/DEMO

If you try this simple experiment, you'll notice a few things. First, loading and saving are incredibly fast. Sure, the

demo program is on the small side, but try a larger program if you like. The speed will amaze you. Meanwhile, the internal mechanical drive stands quietly by the whole time.

Second, all the commands are preceded by /RAM/. In ProDOS terms, /RAM/ is the volume directory for the RAM disk. When you format a blank floppy disk with the *Utilities* program, you assign the disk a volume label, a name for the disk. The format program automatically places a volume directory on the disk, and the volume directory is given the same name as the disk's volume label. The RAM disk, on the other hand, is a permanent part of your computer. You can have only one RAM disk and you don't need to format it. So it makes sense for everybody to use the same volume directory for the RAM disk.

Moving Programs to /RAM/

If you want to move programs from a floppy disk to the RAM disk, all you have to do is type a series of LOAD and SAVE commands. After doing this a few times, however, you'll quickly realize that it's a tedious process.

There *is* a better way. You can use the computer to move the files for you. After all, computers are supposed to be labor-saving devices, aren't they? The solution is a special program which automatically copies your files from the floppy disk to the RAM disk.

Before going further, there are some sticky details that should be resolved.

- The /RAM/ volume directory holds a maximum of 12 entries. How can a larger number of files be handled?
- How will a copy program know which files to move to /RAM/?
- How will a copy program move a program? If it executes a LOAD, won't the copy program itself be destroyed?

The first and second questions can be answered together. All ProDOS volume directories have a limit on the number of entries they can hold. A floppy disk volume directory can hold up to 51 entries, while the /RAM/ volume directory is limited to 12. Fortunately, ProDOS provides a *subdirectory* feature to overcome these restrictions. One or more of the entries in a volume directory can be a directory itself or, in other words, a subdirectory. The subdirectory can have as many entries as you need.

Subdirectories have another important advantage. They're a great way to organize information. You can keep all programs in one subdirectory and data files in another. This is almost like having an invisible barrier on your disk; programs will be in one part and data in another.

Naming the Subdirectories

The RAM disk subdirectory for programs will be called:

/RAM/PROGRAMS/

It would be very convenient if all the programs on floppy disk were also in a subdirectory because the copy program would always know where to find them. Let's establish a standard convention. All BASIC programs on disk will be stored in the subdirectory:

***/XXX*/PROGRAMS/**

where *XXX* represents the volume directory of the disk.

Moving the programs presents a real challenge. With the LOAD command, the incoming program *would* destroy the copy program. One way of solving this is to use another ProDOS feature—EXEC files.

The EXEC Approach

An EXEC file is a text file that contains a series of ProDOS or BASIC commands. The ProDOS and BASIC commands can even be mixed in the same EXEC file. You run the EXEC file by typing

EXEC ***filename***

or

- ***filename***

(The dash is a special command that tells ProDOS to run the program specified by *filename*. Dash is general-purpose. It will run a BASIC or binary program, or execute the commands in an EXEC file.)

The task of moving programs from floppy disk to RAM disk can be performed by an EXEC file with a series of LOAD and SAVE commands:

LOAD /*XXX*/PROGRAMS/PROGRAM1
SAVE /RAM/PROGRAMS/PROGRAM1
LOAD /*XXX*/PROGRAMS/PROGRAM2
SAVE /RAM/PROGRAMS/PROGRAM2

where *XXX* is the floppy disk volume directory.

These commands can be repeated as many times as it takes to copy all the programs.

Creating Subdirectories

You might be wondering how to set up the PROGRAMS subdirectory. You have two choices with floppy disks. When you format a disk with the *Utilities* program, it has provisions for establishing subdirectories. It's a good idea to establish a PROGRAMS and a DATA subdirectory on each disk you format.

You can also use the ProDOS command:

CREATE /*XXX*/PROGRAMS

to create a subdirectory. (Again, *XXX* represents the disk volume directory.)

This second method is also used to set up subdirectories on the RAM disk. CREATE can be a direct command, or it can be executed from within a program.

RAM Disk Mover

All this discussion is a bit on the theoretical side—it's time to get practical. "RAM Disk Mover" is a program which automates the processes already described. It adds a few little twists, however. Here's how RAM Disk Mover works:

- First, it looks for the PROGRAMS subdirectory on your floppy disk. If PROGRAMS is not found, RAM Disk Mover instructs you to insert another disk.
- Next, it looks in the RAM drive for the PROGRAMS subdirectory. If PROGRAMS is not found, RAM Disk Mover creates the subdirectory. If PROGRAMS is found, it deletes all the files in PROGRAMS. This makes room for the new programs.
- RAM Disk Mover goes back to the floppy PROGRAMS subdirectory. Then it saves the name and length (in blocks) of each BASIC program. It stops when there are no more BASIC programs or when the number of blocks exceeds the capacity of the RAM drive (118 blocks, considering the directories).
- Using this list of BASIC programs, RAM Disk Mover builds an EXEC file containing a series of LOAD and SAVE commands. The EXEC file is named TEMP.EXEC and is stored in

the volume directory of the floppy with which RAM Disk Mover is currently working.

- RAM Disk Mover adds a RUN command as the last line of the EXEC file. The program specified by the variable P1$ will start automatically when the copy operation is finished. Right now, P1$ is set up for STARTUP.RAM. You can change this to whatever program name you want.
- After building the EXEC file, RAM Disk Mover clears the screen and informs you that all is well. Then the EXEC file starts up. At this point, programs are actually moved from floppy disk to RAM disk. Your startup program will run and you'll be in business.

Preparing the Mover

RAM Disk Mover requires little, if any, of your attention while it runs. Your biggest job is to organize your disks so that Mover can access them properly.

Here are some guidelines for trouble-free operation:

- Format some disks so that you'll always have spares handy. Use whatever volume labels suit you.
- As you format the disks, be sure to establish the PROGRAM and DATA subdirectories.
- Place a copy of Mover in the volume directory of each disk. Mover doesn't require much space.
- Place all of the BASIC programs you want Mover to copy in the PROGRAMS subdirectory. Don't forget that the RAM disk has a limit of about 61K, or 120 blocks. If you think you'll exceed this limit, place some of the programs on a second disk.

Putting It to Work

Now for the actual operation. It's simple:

1. Always make sure your IIc is turned on, that ProDOS is ready, and that you're in Applesoft.
2. Place your program disk in the internal disk drive.
3. Type either RUN MOVER or – MOVER. RAM Disk Mover will take over from there. It tells you the name of each program it's copying, then starts the EXEC file. As the EXEC file runs, you'll see a series of open brackets displayed. This means all is well.

4. When the EXEC file is done, the STARTUP.RAM program will begin running if it's present.
5. At this point, you can remove your floppy disk from the disk drive. You can insert a data disk and have the entire 140K free for data storage.

Now What?

RAM Disk Mover has done its job. The BASIC programs have been moved over to the RAM disk. *Now, how do you get to the programs?*

If you want to run a program, you can type

RUN /RAM/PROGRAMS/*program name*

or

- /RAM/PROGRAMS/*program name*

Perhaps typing all this seems a bit tedious. You can use the PREFIX command instead:

PREFIX /RAM/PROGRAMS

Now whenever you want to run a program, you can just type RUN followed by the program name. /RAM/PROGRAMS/ is automatically appended to the front of the name.

A caution is in order, however. Suppose that your program performs some file operations. If the input/output statements do not include a full path name, the prefix will also be applied to your data file commands. This will cause your program to try to read or write to the RAM disk—probably not what you intended. Remember this when using the PREFIX command.

You can also use the usual LOAD and SAVE commands, of course. But this time a warning is in order. Yes, you can recall a program from the RAM disk, work on it, test the revisions, and store it back in the RAM disk. *Just don't forget it's only a RAM disk.* If the power goes, then your program goes with it. If you make important changes to a program, save the new version on a real live disk. The RAM disk is best for programs you just want to run.

Oops!

On occasion, things can go wrong—especially where computers are concerned. Here are a few gotchas—watch out for them.

- RAM Disk Mover is designed for an Apple IIc and ProDOS. It uses the 80-column display capability of the IIc. If the display looks funny, make sure your computer is in the 80-column mode.
- Mover attempts to use all the available RAM drive space. It will clean up /RAM/PROGRAMS, but it's not aware of anything else that you may have in the RAM drive. If you run out of room, ProDOS will tell you about it loudly and clearly.
- There can be problems writing the EXEC file. Do not remove your program disk until RAM Disk Mover is finished with it. Do not write-protect it, either. Finally, leave a few blocks free for Mover's use. That EXEC file has to go somewhere.

Additional Hints

Programs can run other programs. If the programs are in the RAM disk, switching from program to program is almost instantaneous. Here's an example:

```
10 REM PROGRAM1
20 D$=CHR$(4)
30 PRINT "PROGRAM 1 IS RUNNING"
40 PRINT D$;"-RAM/PROGRAMS/PROGRAM2"

10 REM PROGRAM2
20 D$=CHR$(4)
30 PRINT "PROGRAM 2 IS RUNNING"
40 PRINT D$;"-RAM/PROGRAMS/PROGRAM1"
```

How about that! Keep this technique in mind when you're writing that huge program that eats all the available program space. The way out is to think small *and* think RAM drive. ProDOS even has a CHAIN statement that permits variables to be passed between programs. (If you would like to learn more about ProDOS, find a copy of Apple's *BASIC Programming with ProDOS*. It covers all the ProDOS features available from Applesoft BASIC.)

Apple IIc RAM Disk Mover

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
C6 100 REM MOVE /.../PROGRAMS TO /RAM/PROGRAMS (MOVE
       R)
84 110 REM
5D 120 PF$ = "/RAM/PROGRAMS/"
24 130 P1$ = "STARTUP.RAM"
B0 140 RB = 118: REM /RAM BLOCKS AVAILABLE
65 150 MP = 50: REM MAX PROGRAMS
48 160 DIM PN$(MP)
14 170 HOME :D$ = CHR$ (4): PRINT D$;"PR#3"
BE 180 GOSUB 330: REM TITLE
F4 190 GOTO 400: REM OPEN DIRECTORY
5C 200 GOTO 650: REM CREATE /RAM DIRECTORY
AE 210 GOSUB 860: REM MOVE PROGRAMS
87 220 REM
56 230 REM THE EXEC PROGRAM WILL MOVE THE PROGRAMS
8B 240 REM
4F 250 HOME
41 260 VTAB 12: HTAB 1: PRINT "Running the EXEC copy
       program ... "
FA 270 FOR I = 1 TO 1000: NEXT
B4 280 PRINT D$;"EXEC ";FL$
57 290 HOME
F8 300 VTAB 12: HTAB 1: PRINT "Programs are being co
      pied to:";PF$
EF 310 FOR I = 1 TO 1000: NEXT
8F 320 END
8D 330 REM PROGRAM TITLE
77 340 UL$ = "": FOR I = 1 TO 80:UL$ = UL$ + "_": NE
      XT
CE 350 TL$ = "DISK TO /RAM PROGRAM MOVER"
5A 360 HOME : VTAB 1: HTAB 1: PRINT UL$
CE 370 VTAB 3: HTAB (80 - LEN (TL$)) / 2: PRINT TL$
BE 380 VTAB 4: HTAB 1: PRINT UL$
25 390 RETURN
71 400 REM OPEN /.../PROGRAMS DIRECTORY
E3 410 GOSUB 590: REM GET VOLUME LABEL
34 420 DR$ = VL$ + "PROGRAMS/"
F5 430 VTAB 6: HTAB 1: PRINT "Moving programs from "
      ;DR$
EA 440 ONERR GOTO 480
BA 450 PRINT D$;"OPEN ";DR$;",TDIR"
ED 460 POKE 216,0: REM NORMAL ERR
1C 470 GOTO 200
F1 480 POKE 216,0: REM NORMAL ERR
A1 490 CALL - 3288: REM FIX STACK
```

```
86 500 VTAB 8: HTAB 1: PRINT "The directory '/PROGRA
       MS/' is not on the diskette ";VL$
17 510 VTAB 10: HTAB 1: PRINT "Insert the proper dis
       kette."
D6 520 VTAB 12: HTAB 1: PRINT "Press SPACE to contin
       ue.  Press RETURN to stop."
E3 530 GET C$: IF C$ < > " " AND C$ < > CHR$ (13) TH
       EN 530
E1 540 FOR I = 6 TO 12 STEP 2
F5 550 VTAB I: HTAB 1: PRINT SPC( 79); CHR$ (13)
09 560 NEXT
93 570 IF C$ = CHR$ (13) THEN STOP
20 580 GOTO 400
49 590 REM GET VOLUME LABEL
83 600 PRINT D$;"PREFIX /"
3C 610 PRINT D$;"PREFIX"
17 620 INPUT VL$
F9 630 PRINT D$
1E 640 RETURN
DF 650 REM CREATE /RAM/PROGRAMS/
7A 660 VTAB 8: HTAB 1: PRINT "Moving programs to ";P
       F$
72 670 ONERR GOTO 820
92 680 PRINT D$;"OPEN ";PF$;",TDIR"
A0 690 PRINT D$;"READ ";PF$
D3 700 INPUT L1$
D7 710 INPUT L2$
DB 720 INPUT L3$
A1 730 INPUT L4$: IF L4$ = "" THEN 770
94 740 T$ = MID$ (L4$,2,15)
DC 750 NP = NP + 1:PN$(NP) = T$
A2 760 GOTO 730
3C 770 PRINT D$;"CLOSE ";PF$
E8 780 FOR I = 1 TO NP
F3 790 PRINT D$;"DELETE ";PF$;PN$(I)
FF 800 NEXT
15 810 GOTO 210
3B 820 POKE 216,0: REM DIRECTORY NOT PRESENT
99 830 CALL - 3288: REM FIX STACK
C2 840 PRINT D$;"CREATE ";PF$
1D 850 GOTO 210
2E 860 REM MOVE PROGRAMS
A9 870 NP = 0: REM NUMBER OF PROGRAMS
48 880 BC = 0: REM BLOCK COUNT
D2 890 PRINT D$;"READ ";DR$
AC 900 INPUT L1$: REM DIRECTORY NAME
E4 910 INPUT L2$: REM TITLE LINE
F4 920 INPUT L3$: REM BLANK LINE
46 930 INPUT L4$: REM FILE ENTRY
DC 940 IF L4$ = "" THEN 970
```

```
03 950 GOSUB 1000: REM SAVE PGM NAMES
A5 960 GOTO 930
C7 970 PRINT D$;"CLOSE"
A3 980 GOSUB 1160: REM BUILD EXEC FILE
2B 990 RETURN
1F 1000 REM SAVE PGM NAMES
50 1010 IF MID$ (L4$,18,3) < > "BAS" THEN 1150
30 1020 NP = NP + 1
EE 1030 IF NP > MP THEN 1150
EA 1040 BC = BC + VAL ( MID$ (L4$,23,6))
3F 1050 IF BC > RB THEN 1150
D4 1060 T$ = MID$ (L4$,2,15)
55 1070 FOR K = 15 TO 1 STEP - 1
A8 1080 IF MID$ (T$,K,1) < > " " THEN 1100
C5 1090 NEXT
85 1100 PN$(NP) = LEFT$ (T$,K)
96 1110 IF PN$(NP) = P1$ THEN SU = 1
53 1120 VTAB 10: HTAB 1: PRINT SPC( 79); CHR$ (13)
0A 1130 VTAB 10: HTAB 1: PRINT "Copying ";PN$(NP);"
      ..."
C0 1140 FOR K = 1 TO 500: NEXT
E7 1150 RETURN
79 1160 REM BUILD EXEC FILE
67 1170 VTAB 10: HTAB 1: PRINT SPC( 79); CHR$ (13)
B3 1180 VTAB 10: HTAB 1: PRINT "Building EXEC copy p
     rogram ..."
F9 1190 FL$ = "TEMP.EXEC"
45 1200 PRINT D$;"OPEN ";FL$
28 1210 PRINT D$;"CLOSE ";FL$
C7 1220 PRINT D$;"DELETE ";FL$
51 1230 PRINT D$;"OPEN ";FL$
36 1240 PRINT D$;"WRITE ";FL$
6D 1250 FOR I = 1 TO NP
36 1260 PRINT "LOAD ";VL$;"PROGRAMS/";PN$(I)
31 1270 PRINT "SAVE ";PF$;PN$(I)
C5 1280 NEXT
F2 1290 IF SU = 1 THEN PRINT "-";PF$;P1$
55 1300 PRINT "PRINT:PRINT"
1D 1310 PRINT D$;"CLOSE"
DF 1320 RETURN
```

Apple Variable Lister

Paul F. Stuever

This fast machine language utility takes the pain out of debugging BASIC programs by listing the current value of every program variable. You can also make a hardcopy of the variable list. Works with DOS 3.3 or ProDOS.

How many times have you run a program, only to get a message like OVERFLOW ERROR IN 240 or, worse yet, BAD SUBSCRIPT ERROR IN 240? When you list the line in question, it may look something like this:

```
240 A$(X+XZ,2*(B/4=C+1),B/4)=STR$(Z)
```

To locate the error, you'll need to type PRINT X, followed by PRINT XZ, followed by PRINT B, and so on, to find the current value of each variable. This is a slow, tedious way to debug a program, especially when you find that some of these variables were defined with other formulas.

"Apple Variable Lister" takes the drudgery out of such debugging tasks by quickly listing the *current* value of every variable in your program. The program is written in machine language and works on any Apple II–series computer with DOS 3.3 or ProDOS.

You can use this utility even if you don't understand machine language: The BASIC loader program listed below creates the machine language and saves it on your disk. Type in the loader, and save a copy before you run it in case you made a typing error. The program has a checksum to catch errors and identify any lines that have mistakes. If no errors are found, it prints OK and saves the utility with the filename VAR.LIST on your disk as a binary file.

Once this is done, you're ready to use the lister. Enter BLOAD VAR.LIST to load it into memory, followed by HIMEM: 31000 to set the top of memory. You will ordinarily want to do this at the beginning of a programming session. If you want, you could use a two-line program as the HELLO

file of that disk which automatically BLOADs VAR.LIST and sets HIMEM to 31000.

```
100 PRINT CHR$(4);"BLOAD VAR.LIST"
200 HIMEM: 31000
```

To list your variables, simply type CALL 32000 and press Return. The same command can run the routine from within a BASIC program. To make a hardcopy of the variable list, enter PR#1 before using the CALL 32000. Make sure your printer is connected and turned on.

Using the Control-Reset combination to break out of a program disables Variable Lister. To reenable it, you'll have to BLOAD it again, as well as reset the HIMEM.

A Chronological List

Variable Lister displays your program's arrays first, followed by floating-point, string, and integer variables. The variables are displayed in chronological order (the order they're used in the program), not alphabetically. Although Applesoft BASIC allows arrays with up to 88 dimensions and as many elements per dimension as available memory will allow, Variable Lister is more restrictive. For this program, arrays are limited to 3 dimensions and a maximum of 254 elements per dimension. Attempting to list a larger array—for example, one created by DIM A$(500)—crashes the utility.

Note that Variable Lister cannot display a variable until it has actually been used in the program. For instance, consider the following line:

```
10 A$="YES":IF A$="NO" THEN B$="OK"
```

Since the IF condition can never be satisfied, B$ will not appear on the variable list unless the program uses it elsewhere. This is no problem when debugging, since you're interested only in variables that were used up to the time the program crashed. However, to make a complete variable list for permanent documentation, you'll need to run your program until you know that every variable has been used.

Apple Variable Lister

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
39 100 HOME : PRINT "CREATING VAR.LIST": HIMEM: 3100
       0
78 110 X = 32000:TC = 0: PRINT
95 120 Z = 0: FOR A = 0 TO 9
B8 130 READ P: POKE X,P:Z = Z + P
93 140 X = X + 1: IF X > 32601 THEN 200
63 150 NEXT A:TC = TC + Z
32 160 READ A: IF A = Z THEN 120
04 170 PRINT "ERROR IN DATA "
C7 180 PRINT "CHECK LINE # ";X - 31010: STOP
BD 190 :
AB 200 IF TC = 85238 THEN PRINT "OK": PRINT CHR$ (4)
       ;"BSAVE VAR.LIST,A$7D00,L601": END
F8 210 PRINT "ERROR IN DATA "
F1 220 PRINT "MISSING A LINE": STOP
61 1000 DATA 032, 127, 125, 169, 000, 133, 004, 032,
        017, 125, 764
F0 1010 DATA 230, 004, 032, 017, 125, 230, 004, 169,
        141, 032, 984
3B 1020 DATA 237, 253, 165, 106, 133, 236, 165, 105,
        208, 009, 1617
BD 1030 DATA 169, 007, 024, 101, 235, 144, 002, 230,
        236, 133, 1281
32 1040 DATA 235, 165, 236, 197, 108, 144, 009, 240,
        001, 096, 1431
8D 1050 DATA 165, 235, 197, 107, 176, 249, 032, 251,
        126, 228, 1766
6E 1060 DATA 004, 208, 223, 032, 019, 127, 160, 002,
        166, 004, 945
EC 1070 DATA 240, 029, 202, 240, 013, 169, 165, 032,
        237, 253, 1580
F9 1080 DATA 032, 072, 249, 032, 228, 126, 208, 198,
        169, 164, 1478
A1 1090 DATA 032, 237, 253, 032, 173, 126, 032, 188,
        126, 208, 1407
67 1100 DATA 185, 032, 072, 249, 164, 236, 165, 235,
        024, 105, 1467
0D 1110 DATA 002, 144, 001, 200, 032, 249, 234, 032,
        046, 237, 1177
C2 1120 DATA 169, 141, 032, 237, 253, 208, 159, 169,
        000, 133, 1501
FD 1130 DATA 004, 032, 141, 125, 230, 004, 032, 141,
        125, 230, 1064
8B 1140 DATA 004, 169, 141, 032, 237, 253, 165, 107,
        133, 235, 1476
```

```
E5 1150 DATA 165, 108, 208, 011, 165, 237, 024, 101,
        235, 133, 1387
AB 1160 DATA 235, 165, 238, 101, 236, 133, 236, 197,
        110, 240, 1891
BD 1170 DATA 003, 144, 007, 096, 165, 235, 197, 109,
        176, 249, 1381
84 1180 DATA 160, 003, 177, 235, 133, 238, 136, 177,
        235, 133, 1627
4A 1190 DATA 237, 032, 251, 126, 228, 004, 208, 212,
        132, 252, 1682
CE 1200 DATA 132, 251, 132, 250, 160, 004, 177, 235,
        170, 200, 1711
04 1210 DATA 200, 177, 235, 149, 249, 202, 208, 247,
        134, 255, 2056
59 1220 DATA 134, 254, 134, 253, 152, 056, 101, 235,
        133, 235, 1687
38 1230 DATA 169, 000, 101, 236, 133, 236, 032, 019,
        127, 166, 1219
CA 1240 DATA 004, 208, 005, 032, 044, 126, 208, 011,
        202, 208, 1048
AB 1250 DATA  005, 032, 074, 126, 208, 003, 032, 102
      , 126, 164, 872
BE 1260 DATA 253, 166, 254, 165, 255, 200, 196, 250,
        144, 016, 1899
A6 1270 DATA 160, 000, 232, 228, 251, 144, 009, 162,
        000, 024, 1210
89 1280 DATA 105, 001, 197, 252, 176, 009, 132, 253,
        134, 254, 1513
C2 1290 DATA 133, 255, 076, 236, 125, 165, 236, 076,
        167, 125, 1594
A6 1300 DATA 032, 127, 126, 165, 235, 164, 236, 032,
        249, 234, 1600
C8 1310 DATA 032, 046, 237, 169, 141, 032, 237, 253,
        024, 169, 1340
A3 1320 DATA 005, 101, 235, 144, 002, 230, 236, 133,
        235, 096, 1417
0B 1330 DATA 169, 164, 032, 237, 253, 032, 127, 126,
        160, 000, 1300
D5 1340 DATA 032, 173, 126, 152, 056, 101, 235, 133,
        235, 169, 1412
3D 1350 DATA 000, 101, 236, 133, 236, 076, 191, 126,
        169, 165, 1433
C2 1360 DATA 032, 237, 253, 032, 127, 126, 160, 000,
        032, 228, 1227
A7 1370 DATA 126, 024, 165, 235, 105, 002, 144, 002,
        230, 236, 1269
5B 1380 DATA 133, 235, 096, 169, 168, 032, 237, 253,
        165, 253, 1741
87 1390 DATA 032, 034, 127, 165, 251, 240, 024, 169,
        172. 032. 1246
```

```
F8 1400 DATA 237, 253, 165, 254, 032, 034, 127, 165,
        252, 240, 1759
5A 1410 DATA 010, 169, 172, 032, 237, 253, 165, 255,
        032, 034, 1359
35 1420 DATA 127, 169, 169, 032, 237, 253, 076, 072,
        249, 177, 1561
A6 1430 DATA 235, 133, 142, 200, 177, 235, 133, 002,
        200, 177, 1634
9B 1440 DATA 235, 133, 003, 096, 032, 072, 249, 169,
        162, 032, 1183
D5 1450 DATA 237, 253, 166, 142, 240, 018, 160, 000,
        177, 002, 1395
5B 1460 DATA 009, 128, 032, 237, 253, 165, 241, 032,
        168, 252, 1517
6A 1470 DATA 200, 202, 208, 240, 169, 162, 032, 237,
        253, 169, 1872
B1 1480 DATA 141, 076, 237, 253, 177, 235, 133, 158,
        200, 177, 1787
0A 1490 DATA 235, 133, 159, 162, 144, 024, 032, 155,
        235, 032, 1311
E7 1500 DATA 046, 237, 169, 141, 076, 237, 253, 162,
        000, 160, 1481
8C 1510 DATA 001, 177, 235, 016, 001, 232, 009, 128,
        133, 001, 933
27 1520 DATA 136, 177, 235, 016, 001, 232, 009, 128,
        133, 000, 1067
79 1530 DATA 096, 165, 000, 032, 237, 253, 165, 001,
        032, 237, 1218
48 1540 DATA 253, 165, 241, 076, 168, 252, 160, 000,
        162, 000, 1477
1E 1550 DATA 201, 100, 144, 012, 160, 176, 162, 176,
        200, 056, 1387
54 1560 DATA 233, 100, 201, 100, 176, 248, 201, 010,
        144, 010, 1423
D4 1570 DATA 162, 176, 232, 056, 233, 010, 201, 010,
        176, 248, 1504
68 1580 DATA 009, 176, 072, 138, 072, 152, 240, 003,
        032, 237, 1131
09 1590 DATA 253, 104, 240, 003, 032, 237, 253, 104,
        076, 237, 1539
5C 1600 DATA 253, 000, 000, 000, 000, 000, 000, 000,
        000, 000, 253
0C 1610 DATA  000, 000, 000, 000, 000, 000, 000, 000
        , 000, 000,  0
```

Lightning Sort

Russ Gaspard
Translation by Tim Victor

In September 1983, COMPUTE! *magazine published "Ultrasort," and we called it the fastest sorting program ever published for any home computer. It would sort a 1000-element array in less than eight seconds.*

It's been improved. Here's "Lightning Sort," which does the same thing in a breathtaking 2.1 seconds. Add this extraordinarily powerful subroutine to any of your BASIC programs where you need to alphabetize something. For all Apple II–series computers with at least 48K and one disk drive, using either DOS 3.3 or ProDOS.

"Lightning Sort" is fast. Compared with "Ultrasort," a routine written for the Commodore computers, Lightning is up to 400 percent faster. It sorts a 1000-element array in an average of 2.1 seconds, versus 7.8 seconds for Ultrasort. Those few seconds' savings aren't much. But when used on random 4000-element arrays, the routine took an average of 10 seconds, versus 40 seconds for Ultrasort. This kind of speedup can be significant in applications where the sort routine is called repeatedly or in sorting very large arrays.

The time for this type of algorithm to sort an *N*-element array is $T*N*Log_2N$ milliseconds on the average, where T is about 0.21 milliseconds for the modified routine and 0.8 milliseconds for the original. Actual running time depends on the starting order of the array.

Besides speeding up the execution, Lightning Sort is also shorter. The Apple version of Lightning needs only 338 bytes of program storage space on disk.

Lightning Typing

To enter Lightning Sort, you'll use "AppleMLX," COMPUTE! Publications' machine language entry and error-checking program. AppleMLX is in Appendix C of this book, along with its instructions. Make sure you have a copy of AppleMLX on disk before you try to enter Lightning Sort. (AppleMLX is used to type in several programs in this book, so save a copy.)

Load and run AppleMLX, and respond to the starting and ending address prompts with

STARTING ADDRESS? 9400
ENDING ADDRESS? 9557

Lightning Sort, Program 1, is quite short, and you should be able to type it in in one sitting. Save a copy of the program to disk, using the filename LIGHTNING.SORT (if you use a different filename, change lines 20 and 30 in Program 2).

Using Lightning

To use Lightning Sort in your own BASIC programs, you have to do only three things:

BLOAD LIGHTNING.SORT. You can do this either in direct mode, or from within your BASIC program. Type BLOAD LIGHTNING.SORT in direct mode or use PRINT CHR$(4) "BLOAD LIGHTNING.SORT" in a program line.

HIMEM: 36864. Setting HIMEM protects your strings from Lightning Sort. If you omit this, there's the chance that Lightning may overwrite your string storage space and cause your program to "crash."

CALL 37888,N,AA$(K). This calls Lightning Sort and sorts *N* number of array variables named AA$, starting with element K.

(Note: If you're using Lightning Sort under ProDOS, setting the HIMEM may cause problems if you try to do any disk operations—it keeps the operating system from setting up a file buffer. Thus, if your program is accessing the disk, you should cancel the HIMEM with **HIMEM: 38400,** then restore it with **HIMEM: 36864** when the disk I/O is done.)

Lightning Sort will alphabetize the strings you've specified in a matter of seconds.

Fast!

To see just how fast Lightning Sort is, type in Program 2, make sure it's on the same disk as LIGHTNING.SORT, and run it.

Program 2 BLOADs LIGHTNING.SORT, sets the HIMEM, and calls the routine. It also generates 1000 random strings and waits for you to press any key. Watch how quickly the message DONE appears on the screen. The 1000 strings are alphabetized. Just to make sure, take a look at them on the screen by hitting any key.

Program 1. Lightning Sort

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program.

```
START ADDRESS: 9400
END ADDRESS:   9557

9400: 20 B1 00 20 05 E1 A5 A0 43
9408: 85 FE A5 A1 85 FD 20 B1 98
9410: 00 20 E3 DF A2 01 A5 83 A3
9418: 9D 52 95 9D 7A 95 A5 84 2B
9420: 9D 66 95 9D 8E 95 A5 FD 52
9428: D0 02 C6 FE C6 FD A0 03 75
9430: 18 BD 7A 95 65 FD 9D 7A 56
9438: 95 BD 8E 95 65 FE 9D 8E B7
9440: 95 88 D0 EC BD 52 95 85 27
9448: 1C BD 66 95 85 1D BD 7A AB
9450: 95 85 1E BD 8E 95 85 1F 3A
9458: 20 12 95 90 04 CA D0 E4 A3
9460: 60 A5 1E 85 1A A5 1F 85 6A
9468: 1B A0 02 B1 1A 99 FA 00 CF
9470: 88 10 F8 30 0B 18 A5 1C 24
9478: 69 03 85 1C 90 02 E6 1D 01
9480: A0 02 B1 1C 99 ED 00 88 7F
9488: 10 F8 20 1D 95 90 E6 38 C2
9490: A5 1E E9 03 85 1E B0 02 89
9498: C6 1F 20 12 95 B0 1F A0 60
94A0: 02 B1 1E 99 ED 00 88 10 25
94A8: F8 20 1D 95 B0 E1 A0 02 A3
94B0: B1 1C 91 1E B9 ED 00 91 E4
94B8: 1C 88 10 F4 30 B7 A0 02 07
94C0: B1 1C 91 1A B9 FA 00 91 E8
94C8: 1C 88 10 F4 18 BD 52 95 65
94D0: 7D 7A 95 85 1E BD 66 95 AC
94D8: 7D 8E 95 85 1F 66 1F 66 A6
94E0: 1E 20 12 95 B0 16 BD 52 68
94E8: 95 9D 53 95 BD 66 95 9D 58
94F0: 67 95 20 32 95 E8 20 42 2D
94F8: 95 4C 44 94 BD 7A 95 9D 72
9500: 7B 95 BD 8E 95 9D 8F 95 C6
9508: 20 42 95 E8 20 32 95 4C 56
9510: 44 94 A5 1D C5 1F D0 04 59
9518: A5 1C C5 1E 60 A0 FF C8 06
9520: C4 ED B0 0B C4 FA B0 06 69
9528: B1 EE D1 FB F0 F1 60 C4 B6
9530: FA 60 18 A5 1C 69 03 9D 78
9538: 52 95 A5 1D 69 00 9D 66 65
9540: 95 60 38 A5 1C E9 03 9D DB
9548: 7A 95 A5 1D E9 00 9D 8E B5
9550: 95 60 00 00 FF FF 00 00 5E
```

Program 2. Lightning Sort Loader and Demonstration

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
D2 10 HIMEM: 38400: HOME : HTAB 8: PRINT "APPLE LIGH
      TNING SORT DEMO"
C8 20 HTAB 10: PRINT "LOADING LIGHTNING.SORT"
89 30 PRINT CHR$ (4)"BLOAD LIGHTNING.SORT"
F2 40 HIMEM: 36864
F5 50 N = 1000
3E 60 DIM AA$(N)
08 70 HOME : PRINT "CREATING "N" RANDOM STRINGS"
12 80 FOR I = 1 TO N
36 90 VTAB 2: PRINT I
02 100 N1 = INT ( RND (1) * 10 + 1)
E1 110 A$ = ""
25 120 FOR J = 1 TO N1
C9 130 B$ = CHR$ ( INT ( RND (1) * 26 + 65))
74 140 A$ = A$ + B$
68 150 NEXT J
01 160 AA$(I) = A$
EB 170 NEXT I
36 180 PRINT "HIT ANY KEY TO START SORT"
C5 190 GET A$: IF A$ = "" THEN 190
17 200 PRINT "SORTING..." CHR$ (7)
12 210 CALL 37888,N,AA$(1)
E4 220 PRINT "DONE" CHR$ (7)
F3 230 PRINT "HIT ANY KEY TO PRINT SORTED STRINGS"
2C 240 GET A$: IF A$ = "" THEN 240
32 250 FOR I = 1 TO N: PRINT I,AA$(I): NEXT
```

Applesoft Lister

David Dobrin

"Applesoft Lister" will give you more-readable program listings, along with printer-oriented output, translated control characters, and indention of nested FOR-NEXT loops. For all Apple II–series computers, with either DOS 3.3 or ProDOS.

Would you like your Applesoft programs to look like this:

```
10 REM  BASIC LISTING WITH APPLESOFT BASIC
20 HOME
22 PRINT "ANT SCRAM"
30 FOR J=0 TO 35
31     VTAB 2
       HTAB J+1
40     PRINT " ;=;@"
50     NEXT
60 PRINT "THAT IS ALL"
```

instead of this?

```
10 REM  BASIC LISTING WITH APPLESOFT BASIC
20 HOME
22 PRINT "ANT SCRAM"
30 FOR J = 0 TO 35
31 VTAB 2: HTAB J + 1
40 PRINT " ;=;@"
50 NEXT
60 PRINT "THAT IS ALL"
```

Applesoft programs are usually difficult to read. The standard LIST function built into Applesoft is unsophisticated, having only the minimum logic necessary to list programs. Here's a program for the Apple that will list Applesoft programs in a nicely formatted fashion. Five major features distinguish "Applesoft Lister" from the standard format:

- There's intelligent spacing between keywords, variables, and operands.
- Multiple statements with a single line number are listed one per line.
- FOR-NEXT constructs are nested.

- Output is oriented for a printer. This listing will not simply "wrap" when it runs out of space on a line.
- Control characters are shown with printable characters.

How Applesoft Lister Works

The program translates the Applesoft intermediate language (IL) into statement numbers and keywords. The keywords are taken from ROM at $D0D0. If this program is to be used with Applesoft in RAM, this value must be changed.

The high byte of the keyword table address is at location $812C. The low byte is at $8130.

When a colon (:) is encountered in the text, the lister starts a new line, indenting appropriately. No action is taken on colons inside double quotes or REM statements. FORs and NEXTs are observed to calculate a nest level.

If you would like to change the indentation of your FOR-NEXT constructs or multiple statements, you can change the value at location 32771 with a POKE statement. POKEing 32771 with 0 turns indenting off, a 3 indents three spaces per nest level, a 10 indents ten spaces per nest level, and so on.

If you want to change the column width, change the value at location 32772 with another POKE. POKE 32772,39 gives a screen width. You can also use 80, 132, or whatever your printer width is.

These POKEs can, of course, be made permanent by saving the program to disk after changing.

Control characters are printed inside brackets; for example, Control-G appears as [G].

Loading the Program into Your Apple

Though the lister program is written entirely in machine language, you don't have to know machine language in order to use it. By using "AppleMLX," COMPUTE! Publications' machine language entry program, you can insure that you have a working copy the first time. Make sure you've read Appendix C and have typed in and saved a copy of AppleMLX to disk (you'll use it to enter a number of other programs in this book). Once you've done that, load and run AppleMLX. It will ask you to type in two numbers:

```
STARTING ADDRESS? 8000
ENDING ADDRESS?   820F
```

Enter Applesoft Lister. When you're finished, use AppleMLX's Save command and name the file ALIST. That's all there's to it. You're now ready to use the Lister.

Running Applesoft Lister

After the program has been stored, it can be used by loading the Applesoft program to be listed in the usual manner. ALIST can then be loaded with

BLOAD ALIST

The listing program can then be run by typing

PR#*x* (where *x* is the slot for your printer interface, if you want the output to go to a printer)

CALL 32768

Note: If you want to change the indentation or width parameters, BLOAD ALIST, make the appropriate POKEs, and save the new version back to disk with

BSAVE ALIST,A$8000,L$F0

Applesoft Lister

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program.

```
START ADDRESS: 8000
END ADDRESS:   820F

8000: 4C 05 80 03 50 A9 00 85 57
8008: 0A A9 01 85 00 A9 08 85 2D
8010: 01 A9 8D 20 9D B1 20 60 43
8018: B1 85 02 20 60 B1 85 03 94
8020: 05 02 D0 01 60 20 60 B1 14
8028: 85 04 20 60 B1 85 05 A9 CC
8030: 00 85 06 85 07 85 08 A2 AC
8038: 10 18 F8 A5 06 65 06 85 18
8040: 06 A5 07 65 07 85 07 A5 E6
8048: 08 65 08 85 08 D8 06 04 B3
8050: 26 05 90 02 E6 06 CA D0 8D
8058: E0 A2 05 A0 00 A5 08 29 EC
8060: 0F D0 0C C0 00 D0 08 E0 DE
8068: 01 F0 04 A9 A0 D0 04 A0 32
8070: 01 09 B0 20 9D B1 98 48 B8
8078: A0 04 06 06 26 07 26 08 8D
8080: 88 D0 F7 68 A8 CA D0 D5 67
8088: A9 A0 20 9D B1 20 6B B1 49
8090: A9 00 85 09 20 60 B1 C9 F6
```

```
8098: 00 D0 10 A9 8D 20 9D 81 14
80A0: A5 02 85 00 A5 03 85 01 EA
80A8: 4C 16 80 A6 0E EC 04 80 7C
80B0: 30 12 48 A2 00 BD 95 81 25
80B8: F0 06 20 9D 81 E8 D0 F5 D8
80C0: 20 6B 81 68 C9 22 D0 08 E3
80C8: A5 09 49 80 85 09 A9 22 D5
80D0: A6 09 D0 1B C9 3A D0 13 1F
80D8: A2 00 BD 8D 81 F0 06 20 B7
80E0: 9D 81 E8 D0 F5 20 6B 81 C3
80E8: 4C 94 80 C9 80 10 1A 29 83
80F0: 7F C9 20 10 0E 48 A9 5B 69
80F8: 20 9D 81 68 09 40 20 9D 4F
8100: 81 A9 5D 20 9D 81 4C 94 FB
8108: 80 48 C9 81 D0 02 E6 0A 15
8110: C9 82 D0 02 C6 0A C9 B2 77
8118: D0 02 E6 09 AA BC 25 81 85
8120: 84 0B 24 0B 10 05 A9 A0 E5
8128: 20 9D 81 A9 D0 85 0D A9 CD
8130: D0 85 0C 68 AA A0 00 CA A7
8138: 10 10 B1 0C E6 0C D0 02 49
8140: E6 0D C9 80 10 F1 30 F2 D6
8148: A0 00 B1 0C C8 AA 20 9D 61
8150: 81 8A 10 F6 24 0B 50 05 1B
8158: A9 A0 20 9D 81 4C 94 80 1D
8160: A0 00 B1 00 E6 00 D0 02 C4
8168: E6 01 60 A2 0D 86 0E A6 9A
8170: 0A 10 02 A2 00 E0 06 30 A6
8178: 02 A2 06 CA 30 0E AC 03 A8
8180: 80 88 30 F7 A9 A0 20 9D 19
8188: 81 4C 81 81 60 8D A0 A0 C2
8190: A0 A0 A0 BA 00 8D A0 A0 E3
8198: A0 A0 A0 A0 00 09 80 20 77
81A0: ED FD E6 0E 60 40 40 40 9C
81A8: 40 40 40 40 40 40 40 00 6B
81B0: 00 40 40 40 40 40 40 00 53
81B8: 40 40 40 40 40 00 00 40 3A
81C0: 40 40 40 40 40 00 40 40 C2
81C8: 00 00 40 40 40 40 00 40 1B
81D0: 40 40 40 40 00 40 40 00 91
81D8: 40 40 40 40 40 40 40 40 DB
81E0: 40 40 40 40 40 00 C0 00 A3
81E8: 00 C0 C0 40 C0 00 00 00 3E
81F0: 00 00 C0 C0 00 00 00 00 18
81F8: 00 00 00 00 00 00 00 00 FB
8200: 00 00 00 00 00 00 00 00 05
8208: 00 00 00 00 00 00 00 00 0D
```

Softsearcher

Ilan Reuben

Here's a short, handy, and fast programming utility written entirely in machine language. With it, you can instantly locate key statements and phrases in your programs. It works on any Apple with at least 48K RAM and a disk drive. You can enter the utility in BASIC or through the machine language monitor. Works under DOS 3.3 or ProDOS.

Many BASIC programs are constructed and debugged by adding new sections and routines to existing sections and routines. As a result, programs can become long and complex. Debugging becomes a real mess when you have to sift through 2000 lines of BASIC to find a certain routine or statement.

"Softsearcher" is a machine language utility which scans any BASIC program for all references to a specified character or phrase, and tells you where each reference is—all in the blink of an eye. The machine language program itself is just over a page (256 bytes) in length and resides at memory location 36864 ($9000 in hexadecimal). If you know little or nothing about machine language, don't worry; you can use Softsearcher as long as you follow a few simple directions.

Using Softsearcher

First, let's get Softsearcher up and running. To make it as painless as possible, Softsearcher has been listed in "AppleMLX" form. AppleMLX, an error-checking machine language entry program, can be found in Appendix C. Make sure you have a copy of AppleMLX on disk and have read its instructions before you begin typing in Softsearcher. (You'll need AppleMLX to enter a number of other programs in this book, as well as some of the Apple programs published in *COMPUTE!* magazine and *COMPUTE!'s Apple Applications* issues, so save a copy.)

Load and run AppleMLX. When it asks for the starting and ending addresses for Softsearcher, type:

```
STARTING ADDRESS? 9000
ENDING ADDRESS?   910F
```

Just start entering the hexadecimal values you see in the listing at the end of this article (Program 1). When you're finished, use AppleMLX's Save command to save Softsearcher to disk. You can name it whatever you want, but for demonstration purposes, we'll assume it's called SEARCHER.

Since AppleMLX creates a machine language file on disk (in the form of a binary file), load the program by typing

BLOAD SEARCHER

Once you have it in memory, you must set the & vector to the start of the program. This lets you run Softsearcher every time you type the character &. From BASIC, enter

POKE 1014,0: POKE 1015,144

Softsearcher should now be ready to use. Here's a sample BASIC program to show how it works. Just type this in (it's not necessary that you run it, although that won't affect Softsearcher's functions).

```
10 PRINT "THIS IS A TEST"
20 FOR A = 1 TO 10
30 PRINT A + 10
40 NEXT A
```

Suppose you want to find all the references to the variable A in the program. You would type

& A

and the computer would respond with

FOUND AT LINE 10

FOUND AT LINE 20

FOUND AT LINE 30

FOUND AT LINE 40

To find all the lines in which the number 10 appears, enter

& 10

Softsearcher hunts through the program and reports

FOUND AT LINE 20

FOUND AT LINE 30

Notice that line 10 was not included, even though there's a 10 in its line number. Softsearcher ignores line numbers.

Note, too, that if you're using Softsearcher on the Apple IIc, it will find characters in lowercase as well as in uppercase. It doesn't matter whether the search characters or phrase is in uppercase or lowercase.

Selective Searching

To specify a range of lines for Softsearcher to look through, type the # character after the &, along with the starting and ending line numbers and the phrase to search for. It could look like this:

```
& #20,30,PRINT
```

This searches lines 20 through 30 for the word *PRINT*. You'll see the report:

```
FOUND AT LINE 30
```

Softsearcher can be used *only in direct mode,* not in deferred mode (that is, you cannot call it from a BASIC program). If you try, the message ?NOT DEFERRED COMMAND ERROR appears.

There are some things you cannot search for. If you enter

```
&&
```

nothing will appear, even if the & symbol was used in a PRINT statement. You also cannot search for the # symbol. If you try, you'll get a SYNTAX ERROR message.

If you'd like to have Softsearcher ready to use every time you boot your system, type in the BASIC setup routine (Program 2) and use it as a HELLO program when initializing disks. Just make sure that you've got the machine language for Softsearcher saved on that disk. It must have been saved as SEARCHER. If it wasn't, make sure that you change lines 20 and 50 in Program 2 to match the new name you've given it.

Program 1. Softsearcher

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program.

```
START ADDRESS: 9000
END ADDRESS:   910F

9000: A5 B9 C9 02 F0 0B A9 0F D1
9008: 20 CC 90 20 19 ED 4C 3C D5
9010: D4 20 B7 00 C9 23 D0 28 3F
9018: 20 B1 00 20 67 DD 20 52 FC
```

```
9020: E7 A5 50 85 08 A5 51 85 FF
9028: 09 20 BE DE 20 67 DD 20 16
9030: 52 E7 A5 50 85 0A A5 51 1F
9038: 85 0B 20 BE DE 4C 4B 90 1E
9040: A0 00 84 08 84 09 88 84 A0
9048: 0A 84 0B A0 FF C6 B8 20 A7
9050: B1 00 C9 22 D0 08 A5 C1 59
9058: 49 E9 85 C1 A9 22 C8 99 66
9060: 0A 91 C9 00 D0 E9 84 06 61
9068: A9 EF 85 C1 A5 08 85 50 CF
9070: A5 09 85 51 20 1A D6 A9 2D
9078: 03 85 07 E6 07 A4 07 A2 47
9080: 00 B1 9B F0 1B DD 0A 91 86
9088: D0 F1 C8 E8 E4 06 D0 F1 09
9090: A9 00 20 CC 90 A0 02 B1 14
9098: 9B AA C8 B1 9B 20 24 ED F9
90A0: A0 00 B1 9B 48 C8 B1 9B 66
90A8: 85 9C 68 85 9B B1 9B F0 E4
90B0: 0A A0 03 B1 9B C5 0B F0 75
90B8: 08 90 BC A9 8D 20 F0 FD 01
90C0: 60 88 B1 9B C5 0A F0 AF 0C
90C8: 90 AD B0 EF AA A9 8D 20 E9
90D0: F0 FD BD DE 90 F0 06 20 04
90D8: F0 FD E8 D0 F5 60 C6 CF AA
90E0: D5 CE C4 A0 C1 D4 A0 CC B2
90E8: C9 CE C5 A0 00 87 BF CE D1
90F0: CF D4 A0 C4 C5 C6 C5 D2 37
90F8: D2 C5 C4 A0 C3 CF CD CD 5E
9100: C1 CE C4 A0 C5 D2 D2 CF 49
9108: D2 00 00 00 FF FF 00 00 94
```

Program 2: Softsearcher (Hello Program)

```
10 D$ = CHR$ (4): REM CTRL-D
20 PRINT D$"BLOAD SEARCHER"
30 POKE 1014,0: POKE 1015,144
40 REM ^ SET & VECTOR ^
50 PRINT "'SEARCHER' ENABLED"
```

Apple Universal Input

William Simpson

Banish EXTRA IGNORED errors from your Applesoft programs with this short INPUT routine. It works on any Apple II–series computer with DOS 3.3 or ProDOS.

If you've ever tried it, you know Applesoft BASIC won't let you type commas or colons when responding to an INPUT prompt. The computer rejects everything after the punctuation and gives you an EXTRA IGNORED error. There's a good reason for this, but there may be times when you'd like an input string to include the punctuation. For example, you might want to input a time value in response to a prompt like

ENTER HOURS:MINUTES.

"Apple Universal Input" solves this problem and can be used as a routine in any Applesoft BASIC program. Once installed, it lets you input strings containing commas and colons, from the keyboard or from disk.

Type in and save the program, enter RUN, and type any string containing commas or colons. The program prints the string to show that the input was accepted without errors.

You'll notice that the input prompt is a greater-than sign (>) rather than a question mark. This signals that the normal Applesoft INPUT command is not in use. If you don't like this prompt, you can easily change it to another character. Find the ASCII code for the character you prefer, add 128 to it, and substitute that value for the second DATA number in line 270 of the program. For example, the less-than symbol (<) has an ASCII code of 60. To use that character as the prompt, you would replace the second DATA number in line 270 with 188 (60 + 128).

Use It Yourself

Let's look at the example program to learn how this input routine can be used in your own programs.

Line 100 defines the variable T$. It's essential that this be the first variable your program defines.

Line 110 POKEs a short machine language (ML) routine into memory; the DATA for this routine is contained in lines 270–300. Lines 120 and 130 print a prompt on the screen and call the new input routine with GOSUB 190. When using this routine in your own programs, you should use a similar GOSUB whenever you want to input a new string. Note that the string is returned in the variable A$ (line 140).

The BASIC subroutine calls the ML routine (CALL 768) to bring the input string into the computer's memory. Using the ROM GETLN routine, the ML routine moves the string into the input buffer, stores the string's length in location 798, subtracts 128 from each character's value to obtain the correct ASCII codes, and returns control to BASIC.

Lines 200–260 move the string from the input buffer to a safe place in memory where it can be accessed by the main program. The vehicle for this transfer is the string variable T$, which you'll recall was the first variable defined in the program. This is done so that you can find the descriptor for T$ by PEEKing the pointer in locations 105–106.

Variable Descriptors

As you may know, a simple variable descriptor consists of five bytes in the following form:

Byte #	Function
1 =	First letter of the variable's name
2 =	Second letter of the name
3 =	Length of the variable
4 =	Low byte of the variable's memory address
5 =	High byte of the variable's memory address

By manipulating the descriptor for the variable T$, it's relatively simple to transfer the string from the input buffer (where it would quickly be overwritten) to another string variable (A$ in example program).

After the descriptor is located (line 210), its third byte is POKEd with the length of the string (line 220), and the fourth and fifth bytes are POKEd with the low byte/high byte address of the input buffer (lines 230–240). T$ is now set to the correct length, and its descriptor points to the input buffer.

The final step (line 250) is to copy T$ into A$, using a form of the MID$ function that extracts every character from T$. You may substitute other names for T$ and A$, of course, when using this routine in your own programs.

Applesoft Universal Input

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
78 100 T$ = ""
F5 110 FOR I = 768 TO 798: READ A: POKE I,A: NEXT
48 120 HOME
1E 130 PRINT "INPUT ANYTHING": GOSUB 190
ED 140 PRINT "ANYTHING==> ";A$
EA 150 PRINT
57 160 INPUT "ANY MORE? (Y OR N) ";YT$
F9 170 IF YT$ = "Y" THEN 120
99 180 END
F2 190 CALL 768
B1 200 B1 = PEEK (798)
A2 210 B2 = PEEK (106) * 256 + PEEK (105)
47 220 POKE B2 + 2,B1
F5 230 POKE B2 + 3,0
1C 240 POKE B2 + 4,2
C1 250 A$ = MID$ (T$,1)
1E 260 RETURN
07 270 DATA  169,190,133,51,32,106,253,142
FF 280 DATA  30,3,164,0,204,30,3,240
76 290 DATA  12,185,0,2,41,127,153,0
D8 300 DATA  2,200,76,12,3,96,0,0
```

Function Keys for the Apple

Ilan Reuben

Frequently used commands (or sequences of commands) can be input more easily by function keys defined by the programmer. This machine language program lets you define up to five function keys for the Apple. For DOS 3.3 only.

A standard feature of many computers (for example, the Commodore 64, the VIC-20, and many Japanese computers) is the function key. It lets you easily enter frequently used commands, or even sequences of commands, and get it right every time. (How many times have you rushed your fingers over the keyboard, only to see you've typed CATALOF or LISY?)

Since the Apple has no special function keys, the only way to get them is through software. "Function Keys" is a machine language utility which provides up to five programmable function keys. A little over half a page (128 bytes) in length, the program resides at location $300 (decimal 768). It also uses a range of memory from $9000 to $94FF. It's simple to enter with "AppleMLX," a machine language error-checking and entry program also included in this book.

Defining Functions

Function Keys allows you to define or use up to five function keys in direct mode. Each function can represent a series of commands of up to 256 characters in length. When you press Control-F, the computer will wait for you to press the number of the function you would like to use (1 through 5), or 0 to define a function. Pressing any other number produces a Control-X.

Once you've indicated what function you want to use, press Return and the computer will act as if you'd typed in the function you used. Pressing the key of an undefined function will just give a carriage return.

To define a function, press Control-F, then press 0. The computer will respond with the message

```
FN#? (1-5,0=EXIT)
```

It's asking you what function number you would like to define. Pressing 0 here aborts the define procedure. When the appropriate function number is pressed, the computer inputs what you want it to record as that function. Type it in as if you were actually giving the computer that command. When you're done, press Return. If you wish to stop while you're typing it in, just press Reset.

Here's a step-by-step example of how to use Function Keys. Suppose you want to turn CATALOG into a function key:

1. Type Control-F.
2. Press 0 since you want to define a function.
3. When you see the computer's prompt (FN#?...), press 1, since you want to define function 1.
4. Type CATALOG, and press Return.
5. Now, whenever you press <Control-F> 1 <Return> you should get a disk catalog.

Putting the Program on Disk

You probably already have a copy of AppleMLX on disk since it's used to enter several other programs in this book. If you don't have a copy yet, refer to Appendix C, where the listing and the program's instructions can be found. Type in AppleMLX and save a copy.

Load and run AppleMLX, and type in the following responses to the two prompts you see on the screen:

```
STARTING ADDRESS? 0300
ENDING ADDRESS?   039F
```

"Function Keys 1.2" (Program 1) is short, and you should be able to type it in in a few minutes. Save it to disk with AppleMLX's Save command, using FUNCTION KEYS 1.2 as the filename. (Remember, Function Keys only works with DOS 3.3, *not* with ProDOS.)

Once you have the machine language program on disk, enter the setup routine (Program 2) and save it. Now, whenever you want to use Function Keys, run the setup program. It's a good idea to use the setup program as part of a HELLO program so that every time you boot that disk, Function Keys will be automatically enabled.

(If for some reason you want to disable Function Keys, enter 9D04:BD 9E from the monitor if DOS is enabled and, if it isn't, enter 36:F0 FD. You might need to disable Function Keys before running certain programs.)

Program 1. Function Keys 1.2

To insure error-free program entry, be sure to use "AppleMLX" (Appendix C) to enter this program.

```
START ADDRESS: 0300
END ADDRESS:   039F

0300: C9 86 D0 1C 18 20 5B 03 63
0308: B0 19 A0 FF C8 C0 FF F0 FA
0310: 42 B1 06 99 00 02 C9 8D 27
0318: D0 F2 84 06 A6 06 CA 60 77
0320: 4C BD 9E A0 10 B9 8B 03 1B
0328: 20 F0 FD 88 10 F7 38 20 B3
0330: 5B 03 A9 F0 8D 04 9D A9 4A
0338: FD 8D 05 9D 20 67 FD A9 5F
0340: 00 8D 04 9D A9 03 8D 05 7D
0348: 9D A0 00 B9 00 02 91 06 12
0350: C8 D0 F8 A2 00 A9 88 8D 7D
0358: 00 02 60 08 20 0C FD C9 62
0360: B6 B0 10 C9 B0 90 0C F0 5D
0368: 18 E9 21 85 07 A9 00 85 D5
0370: 06 28 60 28 A9 87 20 F0 AE
0378: FD A9 98 9D 00 02 68 68 16
0380: 60 28 B0 02 38 60 68 68 73
0388: 4C 53 03 A9 D4 C9 D8 C5 C9
0390: BD B0 AC B5 AD B1 A8 A0 B8
0398: BF A3 CE C6 FF FF 00 00 AD
```

Program 2. Setup Routine

```
10 D$ = CHR$ (4): REM CTRL-D
20 PRINT D$"BLOADFUNCTION KEYS 1.2"
30 OUTVEC = 9 * 4096 + 13 * 256 + 4: REM $9D04
40 POKE OUTVEC,0: POKE OUTVEC + 1,3
50 PRINT : PRINT "FUNCTION KEYS ENABLED."
```

Random Access DATA Statements for the Apple

Robert Jacques Beck

By adding this short routine to your programs, you can gain random access to any piece of information stored in DATA statements—a powerful and useful technique. It works on all Apple II–series computers with either DOS 3.3 or ProDOS.

Any byte in random access memory (RAM) can be immediately accessed during a read or write by specifying its address. *Random access* data files offer the same type of quick access—you locate records by specifying record numbers. Records may be retrieved in any order.

Serial, or *sequential*, access is based on the principle of starting with the first record and counting up to the one you want. Sequential access is usually slower than random access. While it takes about the same time to read any record in an Applesoft random access file, the time required to read an identical record in a sequential file increases as the record is placed toward the file's end. That's because DOS must move past each record in the file to count end-of-record marks until it locates the record it is searching for.

DATA statements in BASIC provide an in-memory sequential access file. You begin by reading the first DATA statement, and you move sequentially through the data list with each successive READ.

DATA statements can be annoying because of this rigidity. They're fine if you want to access your data the same way your DATA statements are organized, but they're difficult to use any other way within the confines of BASIC. Some BASICs use the RESTORE command to reset a pointer to the beginning of the data, but that's not where you always want to go. A few BASICs, such as Atari BASIC, let you RESTORE to a specific line number, providing much more flexibility. But many BASICs (including Applesoft) don't have this feature.

You *can* get flexibility by reading all your DATA statements into arrays and using an index to grab array elements. But storing the same items in DATA statements as well as arrays can eat up memory quickly. Another approach is to read through the data each time until you get to the item you want, using code such as this:

```
10 RESTORE
20 FOR I = 1 TO N
30 READ INFO
40 NEXT I
```

After these lines have been executed, the variable INFO is equal to the *Nth* data item. The major disadvantage of this method is that it's slow.

Flexible Data

Fortunately, there are a couple of zero page pointers that let you manipulate the READ operation. The two short programs included here illustrate how to pull items directly out of DATA statements as if they were in random access files.

In the Apple, decimal locations 123 and 124 (hexadecimal $7B and $7C) store the line number of the last DATA statement read. Locations 125 and 126 point to the item's absolute memory location. The pointers are stored in the usual Apple fashion; that is, the first memory location is the low byte (lower two hexadecimal digits) and the second memory location is the high byte (upper two hexadecimal digits). To translate the information in the pointers into a line number that makes some sense, use this formula:

```
LN = PEEK(123) + PEEK(124)*256
```

Unfortunately, you can't use the line number pointer to do anything. To move from one data item to another, you'll need to adjust the absolute memory pointer. There are a couple of ways to go about it.

Random Languages

Program 1 prints a memory location table of all the items in your DATA statements. Lines 60000 and 60010 print the table's heading. Line 60015 stops the program after the last of the DATA statements is read; line 60020 reads the DATA one

item at a time. Line 60030 calculates the pointer location just after a READ, and line 60040 calculates the current line number. Line 60050 checks to see if the current line number is the same one which was just read; if it isn't, the position index (I = an item's position within a DATA statement) is initialized. Line 60060 prints the table, one row at a time. Just tack these lines onto your program anywhere *after the last DATA statement.* If you use the line numbers from Program 1 (60000–60090), then type RUN 60000 to get your table.

Program 2 is a whimsical little program that shows one way to use the information from Program 1. Lines 70–100 read and print a list of three languages in English. Line 50 reads some memory locations into the array ML. These memory locations were obtained from the table. Pick which language you want the list printed in next. Line 115 sets the variable LOC to the memory location of the appropriate DATA statement. Lines 120 and 130 break the memory location into high and low bytes; then lines 140 and 150 reset the pointer so that the list will be read from the correct DATA statement.

No matter how many times you cycle through the program, the beginning of the list is always printed in the language you want, and you'll never get an END OF DATA message.

Put Program 1 and Program 2 together (you have to delete line 160 of Program 2), and you'll see this table on your screen. Since the locations are calculated *after* a READ, to locate an item, use the value from the immediately preceding item.

Line No.	Position	Location	Item
20	1	2081	ENGLISH
20	2	2089	SPANISH
20	3	2096	FRENCH
30	1	2108	INGLES
30	2	2116	ESPANOL
30	3	2124	FRANCES
40	1	2137	ANGLAIS
40	2	2146	ESPAGNOL
40	3	2155	FRANCAIS

Generated By Combining Program 1 and Program 2

An alternative method is to add or subtract the difference between the pointer's current value and the new value it must have in order to point to an item. Try these changes in Program 2:

```
10 DATA ENGLISH,-32,SPANISH,0,FRENCH,38
20 DATA INGLES,-75,ESPANOL,-38,FRANCES,0
30 DATA ANGLAIS,-114,ESPAGNOL,-82,FRANCAIS,-39
40 REM
50 REM
80 READ A$(I),ML(I)
115 LOC = ML(W) + PEEK(125) + PEEK(126) * 256
```

Line 80 now reads not only the item, but also a number that is added to the pointer in line 115. The advantage here is that we're relying on the separation between items rather than their actual memory locations.

Insert the three DATA statements into the program anywhere you wish. As long as you don't change the relative position of any data, you can edit the program without affecting how the data is handled.

Where the Pointer Points

When a program is run, the pointer is set to the null byte preceding the program's first byte. When the first READ occurs, Applesoft searches for the first line with a DATA token in it (see the Applesoft manual for a description of tokens) and reads everything into a variable until it hits a comma, a zero byte (which signifies end of line), or a colon.

When the next item is read (whether by the same or another READ statement), if the pointer is resting on a zero byte or a colon, Applesoft looks for the next line with a DATA token and starts reading from the byte right after the token. If the pointer rests anywhere else—for instance, on a comma separating two items in a DATA statement—the READ begins immediately after the pointer.

Program 1. Random Access DATA—Table Generator

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in the following two programs.

```
55 60000 PRINT "LINE #" SPC( 3)"POSITION" SPC( 3)"LO
     CATION" SPC( 3)"VARIABLE
8E 60010 FOR I = 1 TO 40: PRINT "-";: NEXT : PRINT
9E 60015 ONERR GOTO 60090
```

```
03 60020 READ A$
D2 60030 LOC = PEEK (125) + PEEK (126) * 256
83 60040 NL = PEEK (123) + PEEK (124) * 256
51 60050 IF NL < > LN THEN I = 1:LN = NL
97 60060 PRINT NL SPC( 10 - LEN ( STR$ (LN)))I SPC(
      10 - LEN ( STR$ (I)))LOC SPC( 11 - LEN ( ST
      R$ (LOC)))A$
40 60070 I = I + 1
C9 60080 GOTO 60020
82 60090 END
```

Program 2. Random Access DATA—Demonstration

```
FB 10 DATA 2068,2096,2124
55 20 DATA ENGLISH,SPANISH,FRENCH
D7 30 DATA INGLES,ESPANOL,FRANCES
82 40 DATA ANGLAIS,ESPAGNOL,FRANCAIS
3D 50 READ ML(1),ML(2),ML(3)
4F 60 HOME
A4 70 FOR I = 1 TO 3
ED 80 READ A$(I)
51 90 PRINT I SPC( 3)A$(I): PRINT
F8 100 NEXT
AC 110 INPUT "WHICH ONE?";W
E8 115 LOC = ML(W)
79 120 HB = INT (LOC / 256)
63 130 LB = LOC - HB * 256
88 140 POKE 125,LB
6C 150 POKE 126,HB
B7 160 GOTO 60
```

Appendices

Guide to Typing In Programs

What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential*, but without a program, it isn't going anywhere. Most of the programs published in this book are written in a computer language called Applesoft BASIC. It's easy to learn and works on the Apple II, II+, IIe, and IIc.

BASIC Programs

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one right way of stating something. Every letter, character, or number is significant. A common mistake is substituting a letter such as *O* for the numeral *0*, a lowercase *l* for the numeral *1*, or an uppercase *B* for the numeral *8*. Also, you must enter all punctuation such as colons and commas just as they appear in the listings. Spacing can be important. To be safe, type in the programs exactly as they appear. Unlike other program listings you may have seen, those in this book have no special characters which you need to interpret. Simply enter the programs as they appear here.

DOS 3.3 and ProDOS

Unless otherwise mentioned in the program's documentation, it doesn't matter whether you have DOS 3.3 or ProDOS. You can enter the programs with either DOS active in your Apple. Of course, you can run a typed-in program only with the DOS system it was entered with.

Uppercase

You'll notice that all the program listings are entirely in uppercase. If you have an Apple IIe or IIc, however, which allows both uppercase and lowercase, you can change text which appears in PRINT statements if you want. A program such as "Home Financial Calculator," for instance, could be modified so that the screen displays appear in both uppercase and lowercase.

DATA Statements

Some programs contain a section or sections of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (called machine language); others contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could lock up, or crash. The keyboard may seem dead, and the screen may go blank. Don't panic—no damage is done. To regain control, you have to turn off your computer, then turn it back on. This will erase whatever program was in memory, *so always save a copy of your program before you run it.* If your computer crashes, you can load the program and look for your mistake.

Sometimes, a mistyped DATA statement will cause an error message when the program is run. The error message may refer to the program line that READs the data. *The error is still in the DATA statements, though.*

Get to Know Your Machine

You should familiarize yourself with your computer before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program so that you won't have to type it in every time you want to use it. Learn to use your machine's editing functions. How do you change a line if you make a mistake? You can always retype the line, but you at least need to know how to use the left- and right-arrow keys. It's all explained in your computer's manuals.

A Quick Review

1. Type in the program a line at a time in order. Press Return at the end of each line. Use the left arrow to correct mistakes.
2. Check the line you've typed against the line in the book. You can check the entire program again if you get an error when you run the program.

Apple Automatic Proofreader

Tim Victor

It's easier than ever to enjoy programs for Apple II–series computers. "Apple Automatic Proofreader," an error-checking program for the Apple II, II+, IIe, and IIc with either DOS 3.3 or ProDOS, alerts you to almost every typing mistake you might make.

"Apple Automatic Proofreader" will help you type in program listings without typing mistakes. It's a short error-checking program that hides itself in memory and attaches to your Apple's operating system. Each time you press Return to enter a program line, this routine displays a two-digit checksum at the top of your screen. If you've typed the line correctly, the checksum on your screen matches the one in the printed listing—it's that simple. You don't have to use the Proofreader to enter listings, but doing so greatly reduces the chance of making a typo.

Getting Started

First, type in the Apple Automatic Proofreader program following this article. The Proofreader can't check itself before it's done, so you'll have to be extra careful to avoid mistakes.

The Proofreader checks which operating system you're running before it hooks up the checksum routine, so you can type it in with either DOS 3.3 or ProDOS. If you want to use the Proofreader with both operating systems, you won't have to retype it. All you need is a utility to copy a file between disks with different formats, such as the one provided on the ProDOS *System Utilities* disk.

As soon as you finish typing the Proofreader, save at least two copies. This is very important, because the Proofreader erases the BASIC portion of itself when you run it, leaving only the machine language portion in memory.

Now, type RUN and hit Return. The Proofreader clears the screen, loads the machine language routine, displays the

message PROOFREADER ACTIVATED, erases the BASIC portion of itself, and ends. If you type LIST and press Return, you'll see that no BASIC program is in memory. The computer is ready for you to type in a new BASIC program.

Entering Programs

Once the Proofreader is activated, you can begin typing in a BASIC program as usual. Every time you finish typing a line and press Return, the Proofreader displays a two-digit checksum number in the upper-left corner of the screen. Compare this checksum with the two-digit checksum printed next to the corresponding line in the program listing. If the numbers match, you can be pretty certain the line was typed correctly. Otherwise, check for your mistake and type the line again.

A common mistake when entering BASIC programs on the Apple occurs when you accidentally press a key while holding down the Control key. This adds an invisible control character to the line you are typing. If you don't find it before you run the program, this stray character may cause a SYNTAX ERROR or other mysterious behavior. Fortunately, the Proofreader detects the presence of these invisible control characters, displaying a checksum that doesn't match the one in the listing. So it's always a good idea to retype a line if the checksums don't match, even though you might not see any difference in the lines themselves.

The Proofreader ignores space characters, so you can omit spaces between keywords and still see a matching checksum. Spaces are important only between the quotation marks of PRINT statements or string assignments. If you accidentally type too many spaces or leave some out, this is the only mistake the Proofreader won't catch. For this reason, you should be extra careful when entering text within quotes.

Before running another BASIC program, it's a good idea to turn off the Proofreader by holding down the Control key while pressing the Reset button. The machine language part of the Proofreader is kept in memory starting at address 768 ($300 hexadecimal). This location is out of BASIC's way, but a lot of other programs use this same place for their machine language subroutines. Disable the Proofreader to avoid conflicts.

How It Works

When the Applesoft BASIC interpreter needs to get a line of input from the keyboard, it calls a machine language routine in the Apple's read only memory (ROM) called GETLN. GETLN, in turn, calls the operating system to get a single keypress, which it stores in an input buffer. If the Return key was pressed, GETLN ends, leaving one new line for the BASIC interpreter in the input buffer. Otherwise, it repeats the process, asking for another keypress.

The operating system normally gets individual keystrokes from a ROM routine called KEYIN, but the Proofreader changes this. When the Proofreader is installed, the operating system calls the checksum routine instead, and the checksum routine asks KEYIN for a character. If any key other than Return was pressed, the checksum routine just passes it on to the operating system, which gives it to GETLN. But if Return *was* pressed, the checksum routine examines the contents of GETLN's input buffer, which now contains an entire line of input, to calculate the checksum that it displays at the top of the screen.

One very common typing mistake is transposition—typing two successive characters in the wrong order, like *PIRNT* instead of *PRINT*. A checksum program that merely adds the codes of the characters in a line can detect only the presence or absence of a character, not transposition errors. Because the Apple Proofreader uses a sophisticated formula to compute checksums, it alerts you to transposed keystrokes.

The Apple Automatic Proofreader detects almost every possible typing mistake, including transpositions, missing or extra characters, accidental control characters, and incorrect line numbers. Typing COMPUTE! Publications' programs into your Apple computer has never been easier.

Apple Automatic Proofreader

```
10 C = 0: FOR I = 768 TO 768 + 68: READ A:C = C + A
   : POKE I,A: NEXT
20 IF C < > 7258 THEN PRINT "ERROR IN PROOFREADER D
   ATA STATEMENTS": END
30 IF PEEK (190 * 256) < > 76 THEN POKE 56,0: POKE
   57,3: CALL 1002: GOTO 50
40 PRINT CHR$ (4);"IN#A$3000"
50 POKE 34,0: HOME : POKE 34,1: VTAB 2: PRINT "PROO
   FREADER INSTALLED"
```

```
60 NEW
100 DATA 216,32,27,253,201,141
110 DATA 208,60,138,72,169,0
120 DATA 72,189,255,1,201,160
130 DATA 240,8,104,10,125,255
140 DATA 1,105,0,72,202,208
150 DATA 238,104,170,41,15,9
160 DATA 48,201,58,144,2,233
170 DATA 57,141,1,4,138,74
180 DATA 74,74,74,41,15,9
190 DATA 48,201,58,144,2,233
200 DATA 57,141,0,4,104,170
210 DATA 169,141,96
```

AppleMLX
Machine Language Entry Program

Tim Victor

Machine language programs are difficult to enter into your computer. To make this chore easier and to eliminate typing mistakes, COMPUTE! Publications presents a machine language entry program for the Apple II, II+, IIe, and IIc computers, using either the DOS 3.3 or ProDOS operating system.

A machine language program is usually listed as a long series of numbers. It's hard to keep your place and even harder to avoid making mistakes as you type in the listing, since an incorrect line looks almost the same as a correct one. To reduce the problems associated with typing in machine language programs, we've presented them as MLX listings which can be entered using the "AppleMLX" editor.

AppleMLX checks your typing on a line-by-line basis. It won't let you enter inappropriate characters, and it won't let you continue if there's a mistake in a line or even if you're trying to enter a line or digit out of sequence. You don't have to know anything about machine language to use it. In other words, AppleMLX makes machine language program entry almost foolproof.

Using AppleMLX

Type in and save AppleMLX to disk (you'll want to use it to enter a number of programs in this book as well as some of those listed in *COMPUTE!* magazine and *COMPUTE!'s Apple Applications* issues). It doesn't matter whether you type it in on a disk formatted for DOS 3.3 or ProDOS. Programs entered with AppleMLX, however, must be saved to a disk formatted with the same operating system as AppleMLX itself.

If you have an Apple IIe or IIc, make sure that the key marked Caps Lock is in the down position. Type RUN. You'll be asked for the starting and ending addresses of the machine language program. These values are given at the beginning of the machine language program listing and in the program's accompanying article. Find them and type them in.

The next thing you'll see is a menu asking you to select a function. The first is (E)NTER DATA. If you're just starting to type in a program, pick this. Press the E key, and the program asks for the address where you want to begin entering data. Type the first number in the first line of the program listing if you're just starting or the line number where you left off if you've already typed in part of a program. Hit the Return key and begin entering the data.

Once you're in Enter mode, AppleMLX will print the address for each program line for you. You then type in all nine numbers on that line, beginning with the first two-digit number after the colon (:). Each line represents eight bytes and a checksum. When you enter a line and hit Return, AppleMLX recalculates the checksum from the eight bytes and the address. If you enter more or less than nine numbers, or if the checksum doesn't exactly match, AppleMLX erases the line you just entered and prompts you again for the same line.

Invalid Characters Banned

AppleMLX is fairly flexible about how you type in the numbers. You can put extra spaces between numbers or leave the spaces out entirely, compressing a line into 18 keypresses. Be careful not to put a space between two digits in the middle of a number. AppleMLX will read two single-digit numbers instead of one two-digit number (F 6 means F *and* 6, not F6).

You can't enter an inappropriate character with AppleMLX. Only the numerals 0–9 and the letters A–F can be typed in. If you press any other key (with some exceptions noted below), nothing happens. This safeguards against entering extraneous characters. Even better, AppleMLX checks for transposed characters. If you're supposed to type in A0 and instead enter 0A, AppleMLX will catch your mistake.

AppleMLX also checks to make sure you're typing in the right line. The address (the number to the left of the colon) is part of the checksum recalculation. If you accidentally skip a line and try to enter incorrect values, AppleMLX won't let you continue. Just make sure you enter the correct starting address; if you don't, you won't be able to enter any of the following lines. AppleMLX will stop you.

Editing Features

AppleMLX also includes some editing features. The left- and right-arrow keys allow you to back up and go forward on the line you're entering so that you can retype data. Pressing the Control key and the D key at the same time (*Delete*) removes the character under the cursor, shortening the line by one character. Pressing the Control key and the I key simultaneously (*Insert*) puts a space under the cursor and shifts the rest of the line to the right, making the line one character longer. If the cursor is at the right end of the line, neither Control-D nor Control-I has any effect.

When you've entered the entire listing (up to the ending address that you specified earlier), AppleMLX automatically leaves Enter mode and redisplays the functions menu. If you want to leave Enter mode before then, press the Return key when AppleMLX prompts you with the address of a new line.

Display Data

The second menu choice, (D)DISPLAY DATA, examines memory and shows the contents in the same format as the program listing. You can use it to check your work or to see how far you've got. When you press the D key, AppleMLX asks you for a starting address. Type in the address of the first line that you want to see and hit Return. AppleMLX displays program lines until you press any key or until it reaches the end of the program.

Save and Load

Two menu selections are provided to let you save programs to disk and to load them back into the computer. These are (S)AVE FILE and (L)OAD FILE. AppleMLX asks you for the name of the file which contains the program. The first time you save a machine language program, there won't be a file on the disk containing the program. Whatever name you type in will be the name of a new file that's created. If the disk doesn't have a file that you requested to load, you'll see a message stating that a disk error has occurred.

If you're not sure why a disk error has occurred, check the disk drive. Make sure that there's a formatted disk in the drive and that it was formatted by the same operating system that you're using for AppleMLX (ProDOS or DOS 3.3). If

you're trying to save a file and see an error message, the disk might be full. Either save the file on another disk or quit AppleMLX (by pressing the Q key), delete an old file or two, then run AppleMLX again.

AppleMLX: Machine Language Entry Program

For mistake-proof entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
8Ø 1ØØ N = 9: HOME : NORMAL : PRINT "APPLE MLX": POK
       E 34,2: ONERR GOTO 61Ø
CC 11Ø VTAB 1: HTAB 2Ø: PRINT "START ADDRESS";: GOSU
       B 53Ø: IF A = Ø THEN PRINT CHR$ (7): GOTO 11Ø
8C 12Ø S = A
E3 13Ø VTAB 2: HTAB 2Ø: PRINT "END ADDRESS  ";: GOSU
       B 53Ø: IF S > = A OR A = Ø THEN PRINT CHR$ (7
       ): GOTO 13Ø
2Ø 14Ø E = A
B5 15Ø PRINT : PRINT "CHOOSE:(E)NTER DATA";: HTAB 22
       : PRINT "(D)ISPLAY DATA": HTAB 8: PRINT "(L)O
       AD FILE  (S)AVE FILE  (Q)UIT": PRINT
AE 16Ø GET A$: FOR I = 1 TO 5: IF A$ < > MID$ ("EDLS
       Q",I,1) THEN NEXT : GOTO 16Ø
93 17Ø ON I GOTO 27Ø,22Ø,18Ø,2ØØ: POKE 34,Ø: END
AF 18Ø INPUT "FILENAME: ";A$: IF A$ < > "" THEN PRIN
       T CHR$ (4);"BLOAD";A$;",A";S
A1 19Ø GOTO 15Ø
6D 2ØØ INPUT "FILENAME: ";A$: IF A$ < > "" THEN PRIN
       T CHR$ (4);"BSAVE";A$;",A";S;",L";E - S
92 21Ø GOTO 15Ø
C2 22Ø GOSUB 59Ø: IF B = Ø THEN 15Ø
9E 23Ø FOR B = B TO E STEP 8:L = 4:A = B: GOSUB 58Ø:
        PRINT A$;": ";:L = 2
85 24Ø FOR F = Ø TO 7:V(F + 1) = PEEK (B + F): NEXT
       : GOSUB 56Ø:V(9) = C
F2 25Ø FOR F = 1 TO N:A = V(F): GOSUB 58Ø: PRINT A$"
        ";: NEXT : PRINT : IF PEEK (49152) < 128 THE
       N NEXT
94 26Ø POKE 49168,Ø: GOTO 15Ø
CC 27Ø GOSUB 59Ø: IF B = Ø THEN 15Ø
48 28Ø FOR B = B TO E STEP 8
A6 29Ø HTAB 1:A = B:L = 4: GOSUB 58Ø: PRINT A$;": ";
       : CALL 64668:A$ = "":P = Ø: GOSUB 33Ø: IF L =
        Ø THEN 15Ø
F9 3ØØ GOSUB 47Ø: IF F < > N THEN PRINT CHR$ (7);: G
       OTO 29Ø
27 31Ø IF N = 9 THEN GOSUB 56Ø: IF C < > V(9) THEN P
       RINT CHR$ (7);: GOTO 29Ø
```

```
72 320 FOR F = 1 TO 8: POKE B + F - 1,V(F): NEXT : P
       RINT : NEXT : GOTO 150
8E 330 IF LEN (A$) = 33 THEN A$ = O$:P = O: PRINT CH
       R$ (7);
22 340 L = LEN (A$):O$ = A$:O = P:L$ = "": IF P > 0
       THEN L$ = LEFT$ (A$,P)
E0 350 R$ = "": IF P < L - 1 THEN R$ = RIGHT$ (A$,L
       - P - 1)
55 360 HTAB 7: PRINT L$;: FLASH : IF P < L THEN PRIN
       T MID$ (A$,P + 1,1);: NORMAL : PRINT R$;
7B 370 PRINT " ";: NORMAL
E6 380 K = PEEK (49152): IF K < 128 THEN 380
C1 390 POKE 49168,0:K = K - 128
5B 400 IF K = 13 THEN HTAB 7: PRINT A$;" ";: RETURN
8A 410 IF K = 32 OR K > 47 AND K < 58 OR K > 64 AND
       K < 71 THEN A$ = L$ + CHR$ (K) + R$:P = P + 1
C1 420 IF K = 4 THEN A$ = L$ + R$
5F 430 IF K = 9 THEN A$ = L$ + " " + MID$ (A$,P + 1,
       1) + R$
0A 440 IF K = 8 THEN P = P - (P > 0)
93 450 IF K = 21 THEN P = P + (P < L)
9D 460 GOTO 330
37 470 F = 1:D = 0: FOR P = 1 TO LEN (A$):C$ = MID$
       (A$,P,1): IF F > N AND C$ < > " " THEN RETURN
BB 480 IF C$ < > " " THEN GOSUB 520:V(F) = J + 16 *
       (D = 1) * V(F):D = D + 1
5F 490 IF D > 0 AND C$ = " " OR D = 2 THEN D = 0:F =
        F + 1
08 500 NEXT : IF D = 0 THEN F = F - 1
17 510 RETURN
B5 520 J = ASC (C$):J = J - 48 - 7 * (J > 64): RETUR
       N
AB 530 A = 0: INPUT A$:A$ = LEFT$ (A$,4): IF LEN (A$
       ) = 0 THEN RETURN
6F 540 FOR P = 1 TO LEN (A$):C$ = MID$ (A$,P,1): IF
       C$ < "0" OR C$ > "9" AND C$ < "A" OR C$ > "Z"
        THEN A = 0: RETURN
2D 550 GOSUB 520:A = A * 16 + J: NEXT : RETURN
28 560 C = INT (B / 256):C = B - 254 * C - 255 * (C
       > 127):C = C - 255 * (C > 255)
20 570 FOR F = 1 TO 8:C = C * 2 - 255 * (C > 127) +
       V(F):C = C - 255 * (C > 255): NEXT : RETURN
DA 580 I = FRE (0):A$ = "": FOR I = 1 TO L:T = INT (
       A / 16):A$ = MID$ ("0123456789ABCDEF",A - 16
       * T + 1,1) + A$:A = T: NEXT : RETURN
1F 590 PRINT "FROM ADDRESS ";: GOSUB 530: IF S > A O
       R E < A OR A = 0 THEN B = 0: RETURN
0D 600 B = S + 8 * INT ((A - S) / 8): RETURN
B6 610 PRINT "DISK ERROR": GOTO 150
```

Index

To order your copy of *COMPUTE!'s Second Book of Apple* Disk, call our toll-free US order line: 1-800-334-0868 (in NC call 919-275-9809) or send your prepaid order to:

COMPUTE!'s Second Book of Apple Disk

COMPUTE! Publications
P.O. Box 5058
Greensboro, NC 27403

All orders must be prepaid (check, charge, or money order). NC residents add 4.5% sales tax.

Send _____ copies of *COMPUTE!'s Second Book of Apple* Disk at $12.95 per copy.

Subtotal $_____

Shipping & Handling: $2.00/disk $_____

Sales tax (if applicable) $_____

Total payment enclosed $_____

All payments must be in U.S. funds.

□ Payment enclosed
Charge □ Visa □ MasterCard □ American Express

Acct. No. _____ Exp. Date _____
(Required)

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 4-5 weeks for delivery.

4580083

COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **800-334-0868** (in NC 919-275-9809) or write COMPUTE! Books, P.O. Box 5058, Greensboro, NC 27403.

Quantity	Title	Price*	Total
_____	Becoming a MacArtist (80-9)	**$17.95**	_____
_____	COMPUTE!'s Apple Games for Kids (91-4)	**$12.95**	_____
_____	COMPUTE!'s First Book of Apple (69-8)	**$12.95**	_____
_____	COMPUTE!'s Guide to Telecomputing on the Apple (76-0)	**$ 9.95**	_____
_____	COMPUTE!'s Kids and the Apple (76-0)	**$12.95**	_____
_____	Easy BASIC Programs for the Apple (88-4)	**$14.95**	_____
_____	MacTalk: Telecomputing on the Macintosh (85-X)	**$12.95**	_____
_____	SpeedScript: The Word Processor for Apple Personal Computers (000)	**$ 9.95**	_____
_____	The Apple IIc: Your First Computer (001)	**$ 9.95**	_____

*Add $2.00 per book for shipping and handling.
Outside US add $5.00 air mail or $2.00 surface mail.

Shipping & handling: $2.00/book _____
Total payment _____

All orders must be prepaid (check, charge, or money order).
All payments must be in US funds.
NC residents add 4.5% sales tax.
☐ Payment enclosed.
Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No._____ Exp. Date _____
(Required)

Signature _____

Name _____

Address _____

City_____ State _____ Zip _____

*Allow 4–5 weeks for delivery.
Prices and availability subject to change.
Current catalog available upon request.

4580083

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!** Magazine. Use this form to order your subscription to **COMPUTE!**.

For Fastest Service
Call Our **Toll-Free** US Order Line
800-334-0868
In NC call 919-275-9809

COMPUTE!
P.O. Box 5058
Greensboro, NC 27403

My computer is:
☐ Commodore 64 ☐ TI-99/4A ☐ Timex/Sinclair ☐ VIC-20 ☐ PET
☐ Radio Shack Color Computer ☐ Apple ☐ Atari ☐ Other _____
☐ Don't yet have one...

☐ $24 One Year US Subscription
☐ $45 Two Year US Subscription
☐ $65 Three Year US Subscription

Subscription rates outside the US:

☐ $30 Canada and Foreign Surface Mail
☐ $65 Foreign Air Delivery

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Payment must be in US funds drawn on a US bank, international money order, or charge card.

☐ Payment Enclosed ☐ Visa
☐ MasterCard ☐ American Express

Acct. No. _____ Expires ____/____
(Required)

Your subscription will begin with the next available issue. Please allow 4–6 weeks for delivery of first issue. Subscription prices subject to change at any time.

758199

Back for More

There are millions of Apple II computers in American homes and businesses. Evidence of this demand is obvious—COMPUTE! Books' *First Book of Apple* has been a best seller since it was released. Now *COMPUTE!'s Second Book of Apple* gives you more excellent, tested, clearly documented programs—more sophisticated games, more thought-provoking educational programs, more useful home applications, more dazzling graphics generators, and more programming utilities.

COMPUTE!'s Second Book of Apple offers a collection of superior Apple programs from past issues of *COMPUTE!* magazine and *COMPUTE!'s Apple Applications*, as well as programs and articles never before published. Here's just a sampling of what you'll find inside:

- "Heat Seeker," a fast-action arcade game that pits your piloting skills against persistent heat-seeking missiles.
- "Bowling Champ" puts you on the alley, going for your first 300 game.
- "Home Financial Calculator" lets you figure almost every loan and investment possibility.
- "QUICK.BUT.DUMB," a simple-to-use terminal program, connects your computer and modem with databases like CompuServe, The Source, Dow Jones News/Retrieval, and hundreds of electronic bulletin boards.
- "Missile Math" teaches addition, subtraction, multiplication, and division while children play a game.
- "Chess," a five-level computer chess opponent.
- "Apple SuperFont," a sophisticated character generator for the Apple hi-res screen.
- "Apple IIc RAM Disk Mover" effectively puts a second disk drive in your Apple IIc.
- "Softsearcher," a utility which finds any string of characters in any program.

Each program is ready to type in and run. Instructions are clearly written and complete.

COMPUTE!'s Second Book of Apple continues the tradition of offering quality programs to Apple owners. It gives you more of what you want—outstanding software for your Apple computer.

ISBN 0-87455-008-4

COMPUTE!'s Second Book of Apple

COMPUTE! Books